سلسلة أساسيات البرمجة: ٤

كتاب منهجي
٢٠٠٦

إعداد المهندس
أحمد جبر عبدربه جحيش

مدخل أساسي إلى البرمجة كائنية التوجه

أساسيات البرمجة بلغة جافا

Fundamentals of Programming in Java

دار إعادة هندسة الفكر

بسم الله الرحمن الرحيم

أساسيات البرمجة
بلغة جافا

Fundamentals of Programming in JAVA

أُعد هذا الكتاب في الأصل كمقرر دراسي في مادة

البرمجة كائنية التوجه (١).

الطبعة الأولى، ٢٠٠٦م

بطاقة فهرسة الكتاب أثناء الإعداد للنشر

جحيش، أحمد جبر عبدربه

أساسيات البرمجة بلغة جافا. / أحمد جبر عبدربه جحيش - صنعاء، ٢٠٠٦م

٢٣٨ ص ؛ ١٧ سم × ٢٤ سم (سلسلة أساسيات البرمجة؛ ٤)

ردمك: 978-xxx-xxx-x

١- لغة جافا ٢- لغات البرمجة ٣- البرمجة ٤- علوم الحاسوب

أ. العنوان ب. السلسلة

ديوي 005.133J ٢٠٠٦/xxx

رقم الإيداع بدار الكتب: ٢٠٠٦/xxx

ردمك: 978-X-XXX-XXX

للتواصل وإبداء الملاحظات والآراء

ahmad.jbr@gmail.com

00967-73549637

أساسيات البرمجة بلغة

جافا

Fundamentals of Programming in

JAVA

إعداد المهندس

أحمد جبر عبدربه جحيش

مدرس البرمجة كائنية التوجه OOP، كلية العلوم والهندسة – جامعة العلوم والتكنولوجيا – صنعاء

مدرس أسس لغات البرمجة Essentials of Programming Languages – الجامعة الوطنية – صنعاء

مؤسس ونائب مدير شركة Skytech للحلول الذكية وتكنولوجيا المعلومات

٢٠٠٦م

بسم الله الرحمن الرحيم

﴿ وَٱبْتَغِ فِيمَآ ءَاتَىٰكَ ٱللَّهُ ٱلدَّارَ ٱلْـَٔاخِرَةَ ۖ وَلَا تَنسَ نَصِيبَكَ مِنَ ٱلدُّنْيَا ۖ
وَأَحْسِن كَمَآ أَحْسَنَ ٱللَّهُ إِلَيْكَ ۖ وَلَا تَبْغِ ٱلْفَسَادَ فِى ٱلْأَرْضِ ۖ
إِنَّ ٱللَّهَ لَا يُحِبُّ ٱلْمُفْسِدِينَ ﴾

القصص: ٧٧

قال رسول الله ﷺ

﴿ مَنْ سَلَكَ طَرِيقًا يَلْتَمِسُ فِيهِ عِلْمًا سَهَّلَ اللَّهُ لَهُ طَرِيقًا إِلَى الْجَنَّةِ ﴾

رواه الترمذي

الإهداء

إلى والدتي حفظها الله

إلى والدي رحمه الله

إلى زوجتي وابنتي ملك

إلى إخوتي وأخواتي ،،،

الشكر والتقدير

أ. فيصل مسعود

أ. أحمد عبد العظيم

أ. عبد الله المصباحي

م. أحمد جبر

المحتويات

مُقَدِّمَة

إن الهدف من هذا الكتاب هو توفير مرجع تعليمي منهجي عربي للمبرمجين وخاصة دارسي البرمجة كائنية التوجه، وهو يسعى إلى تعليمهم أسس البرمجة بلغة جافا، وعلى الرغم من أن الخبرة في استخدام لغة C أو C++ ليست ضرورية؛ إلا أنها ستكون مفيدة، إذ يتعامل الكتاب مع القارئ على أنه يبدأ مجال البرمجة لأول مرة مع لغة جافا، كما لم يخلُ الكتاب من بعض الإشارات هنا وهناك لبعض المقارنات بين لغة جافا وغيرها من اللغات المنتشرة.

ونحن نسعى من خلال هذا الكتاب إلى تغطية مجموعة من المواضيع الأساسية؛ التي يمكن إيرادها في الآتي:

- **المفاهيم الأساسية في لغة جافا؛ وأهم خصائصها ومميزاتها ومكوناتها، مع مدخل مفاهيمي للبرمجة كائنية التوجه.**
- **بناء وتطوير تطبيقات جافا ومقدمة إلى بُريمجيات جافا؛ وإمكانياتها ودورة حياتها وطرق تنفيذها، مع نبذة مختصرة عن استخدام المحرر Kawa.**
- **التعامل مع خرج البرامج وتنسيقه وإدارته، وحجز المواقع في الذاكرة والمتغيرات وأنماطها وخصائصها.**
- **طرق الإسناد وإجراء العمليات الحسابية وعمليات المقارنة وعمليات الزيادة والإنقاص، والعمليات المنطقية.**
- **أساسيات الرسم في لغة جافا.**
- **التعرف على الخوارزميات ولغتها (الشفرة الزائفة) Pseudocode.**
- **نظرة شاملة حول بُنى التحكم الأساسية (بُنى الاختيار والتكرار)، والتعليمتان break وcontinue وإمكانياتهما.**
- **أسلوب البرمجة المهيكلة Structured Programming.**

- نبذة مختصرة عن لغة النمذجة الموحدة UML.
- استخدام الطرق والتوابع الجاهزة والطرق الرياضية المضمنة وبناء وتطوير الطرق والتوابع الخاصة.
- قواعد مجال الرؤية Scope Rules، والتحميل الزائد للطرق Method Overloading، والاستدعاء الذاتي (العودية) Recursion.

ولتحقيق هدف هذا الكتاب فقد اعتمدنا على الأسلوب والمنهجية المقدمة من قبل كتاب «Java How to Program 5th ed.» كمرجع أساسي في المواضيع والأمثلة المقدمة نظرًا لأنه يمثل أقوى المناهج التعليمية في مجاله - حسب علمنا - بالإضافة إلى الاستعانة بالعديد من المراجع العربية والأجنبية لإثراء هذه المادة العلمية.

وقد أُعد هذا الكتاب في الأساس كمقرر دراسي في مادة البرمجة كائنية التوجه (١) لطلبة مستوى ثاني في قسم علوم الحاسوب والمعلوماتية، وطلبة مستوى رابع في قسم الهندسة الإلكترونية.

والكتاب يقدم مدخلاً أساسيًا إلى البرمجة كائنية التوجه، خاصة وأن لغة جافا قد بنيت أساسًا كلغة كائنية، ونظرًا للحاجة للتوسع في هذا الموضوع واقتصار هذا الكتاب على «**أساسيات البرمجة بلغة جافا**»؛ فقد قمنا بإعداد كتاب آخر يمكن اعتباره الجزء الثاني أو المكمل لهذا الكتاب بعنوان: «**أساسيات البرمجة كائنية التوجه باستخدام لغة جافا**».

وأخيرًا، فإننا نأمل أن يستفيد الأخوة الدارسين والمبرمجين من هذه المادة العلمية، وأن يتقبله اللّه خالصًا لوجهه الكريم وأن ينفع به كل طالب علم ومعرفة.

واللّه من وراء القصد ،،،

م. أحمد جبر

الفصل الأول

مقدمة إلى لغة جافا

Introduction to Java

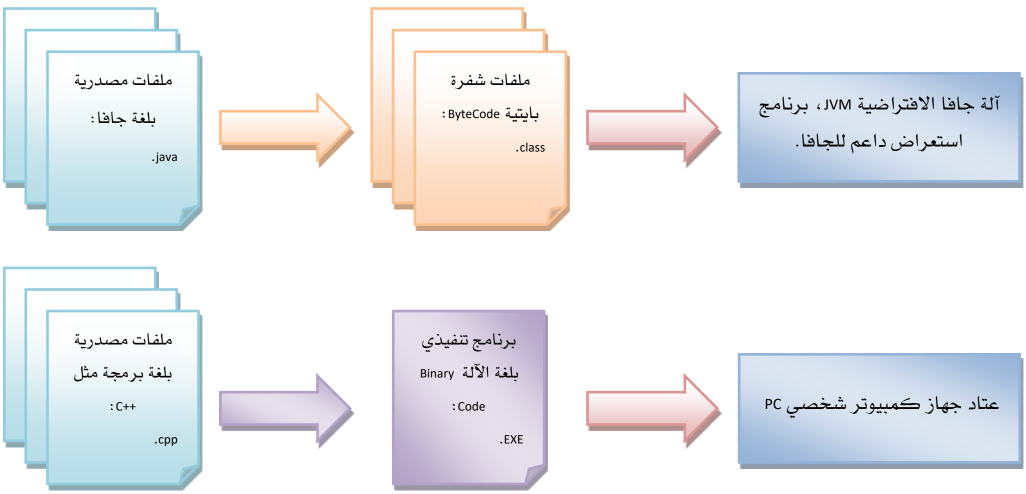
تمهيد:

لغة Java هي من اللغات الحديثة إذ أنتجتها شركة Sun Microsystems عام ١٩٩٥. قامت شركة صن باستخدام Java في العديد من المشروعات لإظهار إمكانياتها الكبيرة، ولكن لم تصبح Java محط الأنظار أو موضع الاهتمام إلا بعد ظهور المستعرض HotJava فقد قامت شركة صن بكتابة HotJava بسرعة لإظهار جودة وقدرة Java على التعامل مع الإنترنت ولتوضيح درجة تعقيد التطبيقات التي يمكن تطويرها. كان الإصدار الثاني من Netscape Navigator (بالنسبة للنظام Windows) هو أول متصفح تجاري يدعم Java 1.0، وتلاه بعد ذلك Internet Explorer. عندما تم استخدام Java على شبكة الويب حققت نجاحًا كبيرًا. صُممت لغة جافا أساسًا كي تكون مستقلة عن محيط التشغيل، وقد عبرت عن ذلك شركة Sun بمبدأ « write once, run everywhere» أي «اكتب لمرة واحدة وشغل أينما كنت».



ويتطلب تجسيد هذا المبدأ تعريف وتحقيق ما نسميه بآلة جافا الافتراضية

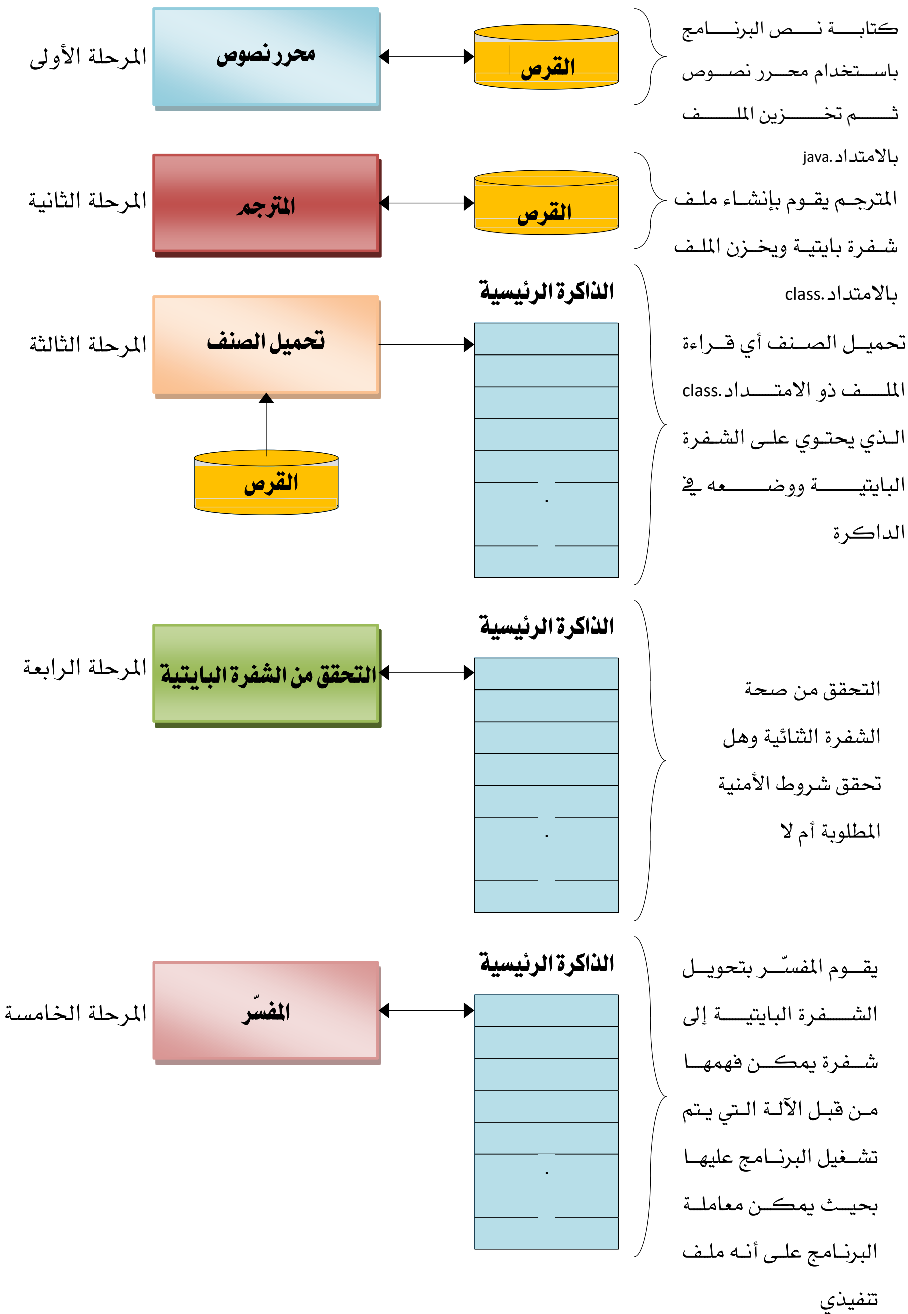
(JVM) Machine Virtual Java، وهـذه الآلـة تحـاكي عمـل معـالج لـه تعليماتـه الخاصـة، وبحيـث يسـمح تحقيـق هـذه الآلـة علـى منصـات عمـل متنوعـة بنقـل بـرامج جافا التنفيذية من محيط تشغيل إلى آخر وتشغيلها دون الحاجة لتعديلها.



ترجمـة ملفـات المصـدر ذات الامتـداد java يـؤدي إلى ملفـات خـرج ذات الامتـداد class تحتـوي علـى تعليمـات شـفرة جافـا البايتيـة ByteCode. ومـن ثـم يقـوم برنـامج الاسـتعراض الـداعم للجافـا بتحميـل تلـك التعليمات وتنفيذها.

أي أنـه عنـد ترجمـة البرنـامج ينـتج ملفـات شـفرة الجافـا البايتيـة ByteCode. إن المـرور على آلة جافا الافتراضية يخدم ثلاثة أهداف:

- **الاستقلالية عن الآلة:** تم تصـميم JVM خاصـة بكـل نـوع كمبيـوترات علـى حـدة ولكـن الهـدف الأساسـي مـن كـل هـذه النسـخ هـو تحويـل الشـفرة البايتيـة إلى شفرة بلغة تلك الآلة التي يعمل البرنامج عليها.



بيئة عمل جافا مثالية

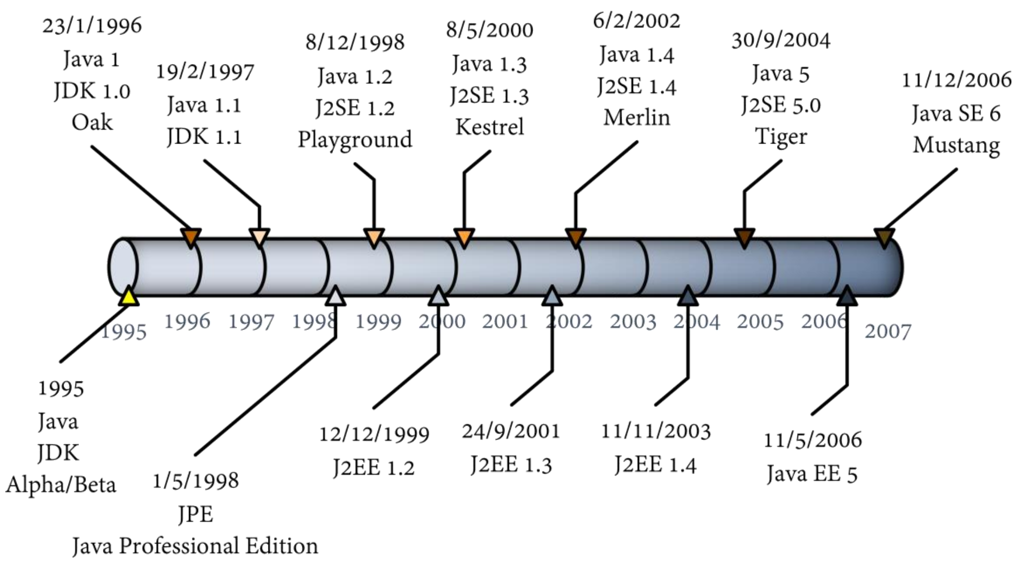
- **الحماية:** لقد أظهرت الخبرة كم من الصعب تحقيق أي مستوى معقول من الحماية للبرامج. طالما كان تنفيذ البرنامج يتم في وحدة المعالجة المركزية CPU في النمط المحلي، فإن أي خطأ في النظام أو برنامج الاستعراض يمكن أن يكشفه أي شخص يبحث عن طريقة للدخول إلى آلة المستخدم. إن برامج الجافا لا تملك هذه المرونة. فبرنامج الجافا يتم تنفيذه ضمن آلة جافا الافتراضية التي أنشأها برنامج الاستعراض. وطيلة فترة تنفيذ البرنامج، يكون برنامج الاستعراض حاضرًا دائمًا ليُشرف على عملية التنفيذ للتأكد من عدم تعرض البرنامج لأي أذى.

- **حجم برامج مخفّض:** إن برامج آلة جافا الافتراضية أصغر بكثير من البرامج التقليدية. والسبب هو أن توابع مكتبات الجافا التي يستدعيها برنامجك متواجدة في برنامج الاستعراض وليست مرتبطة بالبرنامج. يؤدي هذا إلى تخفيض حجم بريمج الجافا إلى حدٍ كبير مما يخفض الوقت المطلوب للتحميل. بالإضافة إلى ذلك، تخفض الجافا وقت التحميل من خلال تحميل الأمور التي تحتاج إليها فقط (لا يتم تحميل الشفرات البايتية التي تولدها فئة ما إلا عندما يحتاج إليها البرنامج).

تلميح: إن لهذه اللغة مثل باقي لغات البرمجة محررات وبرامج تطبق فيها لغة جافا ومنها Java Developer Kit (JDK) وهي منتجة من قبل شركة صن، ++Visual J من مايكروسوفت، Visual Cofee من شركة سيمانتك، Jbuilder من بورلاند، Kawa من تك - تولز، ... وبالتالي يمكنك اختيار البيئة التي ترغب في استخدامها كمحرر أساسي لديك.

الإصدارات المختلفة للجافا:

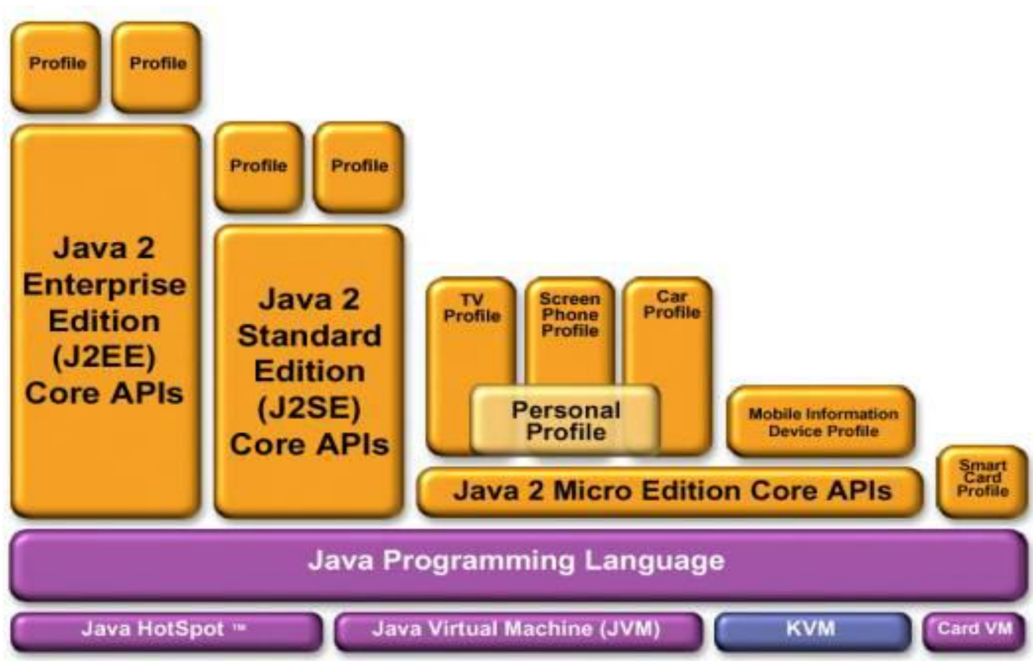
شـأنها شـأن التقنيـات المتميـزة الأخـرى، يحـدث تطـور دائـم في الجافـا. تم إصـدار Java Development Toolkits (JDKs) والـتي سميـت فيمـا بعـد Software Development Toolkit (SDK) الأولى في منتصـف التسـعينات. عـبر الوقـت تطـورت Java بصـورة ملحوظـة مـن بـداياتها المتواضـعة كبيئـة تعمـل علـى العديـد مـن نظـم التشـغيل ويمكـن اسـتخدامها في إنشـاء الـApplets البريمجيـات (بـرامج Java الصـغيرة الـتي يمكـن إدراجهـا مباشـرة في صـفحات الويـب) البسـيطة والتطبيقـات المسـتقلة الأوليـة، إلى أن أصبحت اليوم نظام تشغيل متكامل نادت به شركة صن.



وبـذلك أصـبحت لغـة Java المقيـاس الفعلـي لتطـوير بـرامج الإنترنـت الـتي يسـتخدمها حاليًـا أكثـر مـن سـتة مليـون مـبرمج في أنحـاء العـالم. ظهـرت النسـخة ١,١ مـن Java بعـد الأولى بسـنوات قليلـة وتلتهـا النسـخة ١,٢ والـتي ظهـر معهـا نظـام التشـغيل Java والـذي حمـل الاسـم التجـاري «Java 2 Platform» الإصـدار الثـاني مـن Java في أواخـر عـام

١٩٩٨. ثم تلتها النسخة ١,٣ وصولاً إلى النسخة Java SE 6 وهي أحدث وأفضل نسخة تنفيذية لـJava 2 Platform وتمتلك هذه الإصدارة النوعيات التالية:

- Java 2 Standard Edition (J2SE): تقدم لنا هذه النسخة كل ما نحتاجه لإنشاء بريمجيات وتطبيقات قوية باستخدام الإصدار الثاني من Java لأجهزة الكمبيوتر الشخصية.
- Java 2 Enterprise Edition (J2EE): تستخدم هذه النسخة للتطبيقات المهمة على مستوى الشركات.
- Java 2 Micro Edition (J2ME): صُممت هذه النسخة خصيصًا لاستخدامها مع أجهزة مثل التلفونات الخلوية وغيرها.

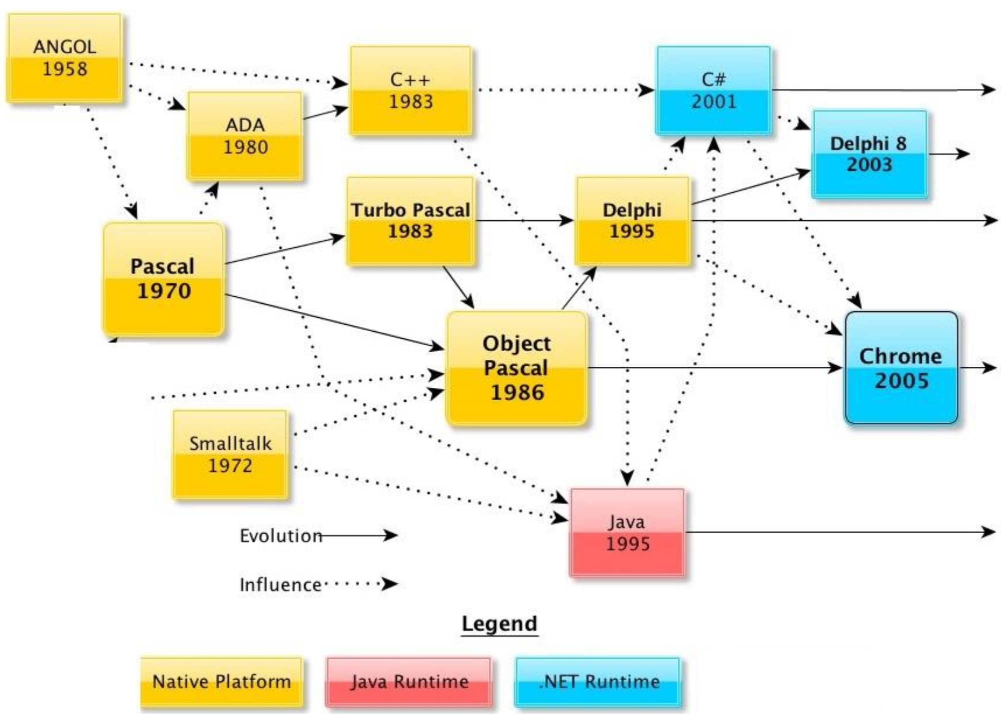


مقدمة عن بيئة Java:

تعتبر Java تقنية جديدة معقدة وقوية، حيث أنها تحتوي على الكثير من الواجهات التي تعمل معًا من أجل إنجاز المهام. وسنورد فيما يلي بعض المفاهيم الأساسية:

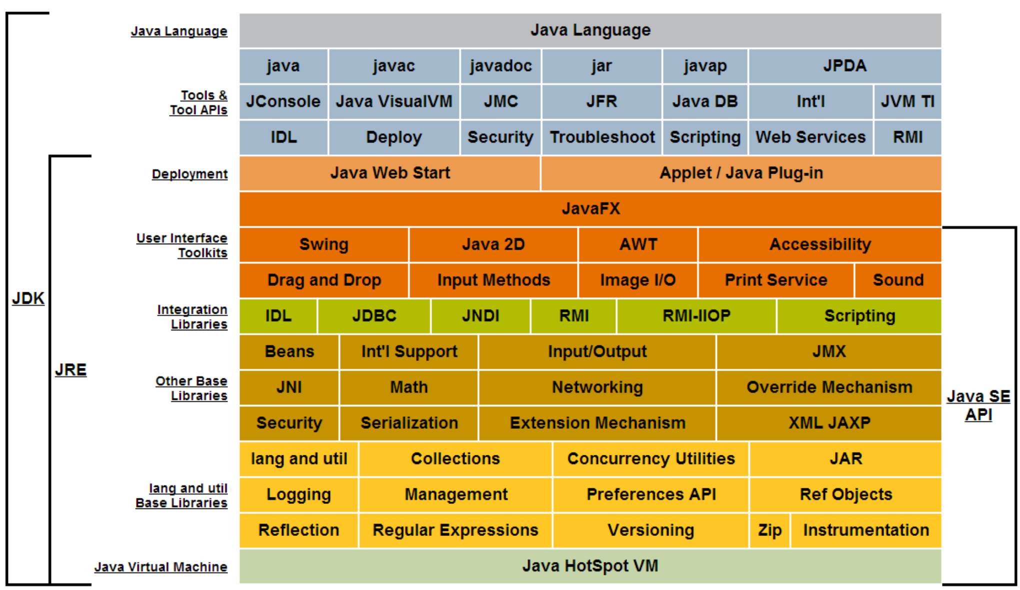
لغة Java:

إن لغة Java هي إحدى لغات البرمجة التي تعتمد على الكائنات وقد طورتها شركة صن. وعلى الرغم من أن لهذه اللغة جذورًا في لغة C++، إلا أنها تعتبر إحدى لغات تطوير البرامج الجديدة. وعلى عكس لغة C++، إن لغة Java تعتمد اعتمادًا كليا على استخدام الكائنات وقد صُممت بحيث تعمل على أي بيئة تشغيل وتنشئ بريمجيات applets وتطبيقات ومعالجات.



مكتبات الفئات (الأصناف) الخاصة بالـJava:

تأتي كل الأنظمة التي تستطيع التعامل مع لغة Java مع مجموعة قياسية من المكتبات التي تحتوي على قوالب الكائنات المنشأة مسبقًا، أو الفئات، والتي يمكن للمبرمجين استخدامها في إنشاء البرامج. ويقدم الإصدار الثاني من Java آلاف الفئات وكذلك مجموعة من قوالب الكود المنشأة مسبقًا والتي يمكنك استخدامها كما هي أو يمكنك تخصيصها (توسيعها) لخدمة الأغراض الخاصة بك. وتتاح مكتبات الفئات القياسية الخاصة بـJava لأي مهمة، بما في ذلك عمل النوافذ وإنشاء واجهة الاستخدام الرسومية والمدخلات/المخرجات والاتصال الشبكي وما إلى ذلك. إن مكتبات الفئات الموسعة الخاصة بـJava تسمى أحيانًا Java Application Programming Interface (API)، لأنها تحدد واجهات Java القياسية التي تكون لدى المبرمجين عند قيامهم بإنشاء البرامج الخاصة بهم.

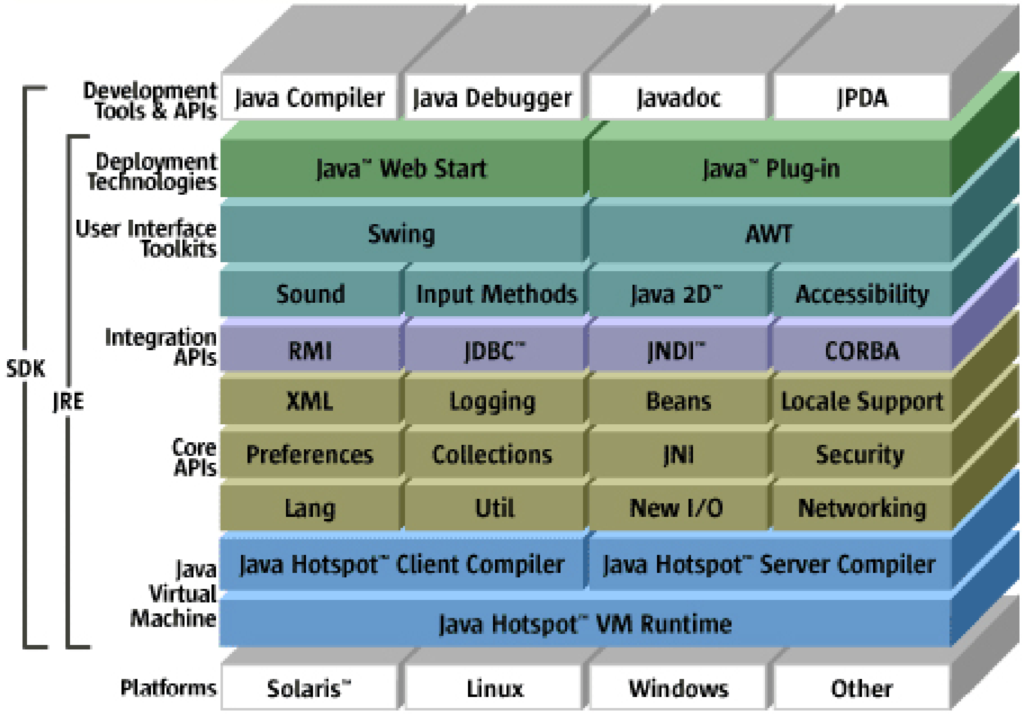


بيئة تشغيل Java:

بعد أن تقوم بكتابة برنامج بلغة Java، فإن تنفيذ الكود يتوقف على بيئة التشغيل. تتألف بيئة التشغيل الخاصة بـJava، والتي تسمى أيضًا هيكل Java، من لغة Java و(JVM) Java Virtual Machine وهما يمثلان معًا الوسيلة لتنفيذ كود Java. يمكن أن تعمل بيئة تشغيل Java على أي نظام تشغيل، وهي أساس إمكانيات المعالجة الموزعة الخاصة بـJava، كما أنها متاحة لأشهر نظم التشغيل وأكثرها شيوعًا. والآن، تتاح بيئة تشغيل Java للمستخدم النهائي من خلال متصفحات الويب مثل متصفح HotJava الذي قدمته شركة Sun ومتصفح Netscape Navigator الذي قدمته شركة Netscape Communications Corp. ومتصفح Internet Explorer الذي قدمته شركة Microsoft. وتسمح مثل هذه المتصفحات المتوافقة مع Java للأشخاص بتجربة نموذج خاص جدًا من Java، وهو Java applet.

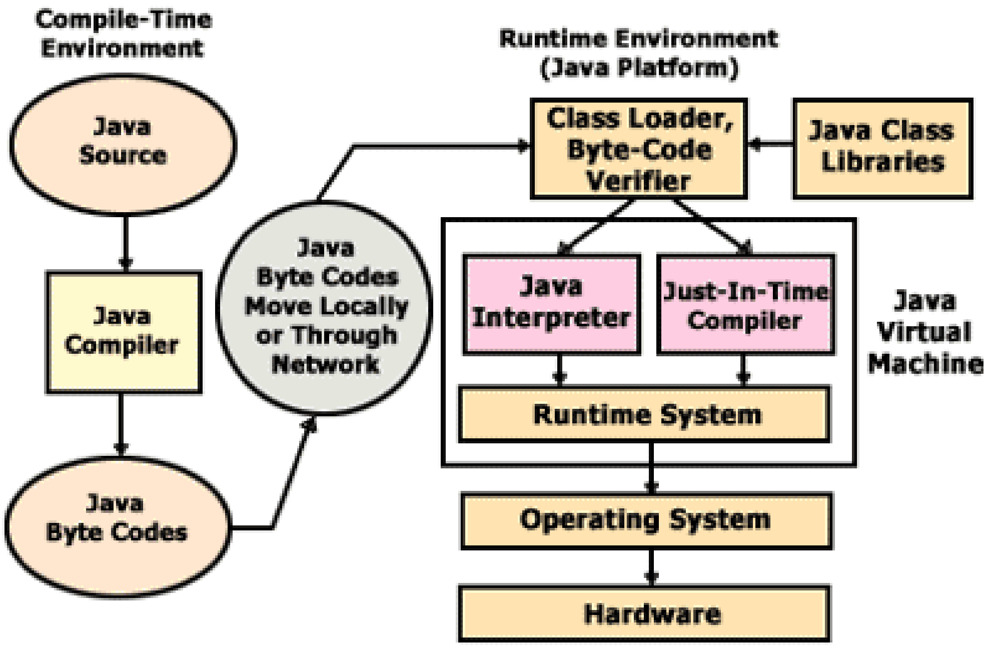
على الرغم من أن استخدام متصفحات الويب المتوافقة مع Java هو أشهر طرق اكتساب الخبرة في تقنية Java، فإن هذه الطريقة لا تعد الطريقة الوحيدة. حيث تتاح بيئات تشغيل مستقلة تسمح للمستخدم النهائي بتشغيل تطبيقات سطح المكتب غير المضمنة في صفحات الويب.

يأتي أحدث نموذج من Java، وهو الإصدار الثاني، مع إحدى بيئات التشغيل القوية جدًا وسوف تستخدمها لتطوير واختبار تطبيقات Java الخاصة بك. وباستخدام (SDK) Java 2 Software Development Toolkit المجانية المقدمة من شركة Sun، سوف يكون بمتناول يديك كل أدوات Java التي تحتاجها لكتابة وتصحيح أخطاء ونشر كل برامج Java.



آلة Java الافتراضية Java Virtual Machine:

إن Java Virtual Machine هي مواصفات تم تحديدها بشكل صحيح ويجب عليك إتباعها عند كتابة كود Java. ويتم تجميع كل كود Java عليها ، وهذه المواصفات خاصة بكيفية إنتاج الكود أثناء عملية تجميع البرنامج، وتساعد كتابة الكود الذي يعمل في بيئة Java Virtual Machine على ضمان العمل على أي نظام تشغيل، على الرغم من ذلك يمكنك أن تضمن عمل الكود على أي نظام تشغيل إذا لم تستخدم أي كود داخلي مرتبط بنظام تشغيل بعينه كما يجب ألا يتضمن الكود عمليات داخلية خاصة بأي نظام.

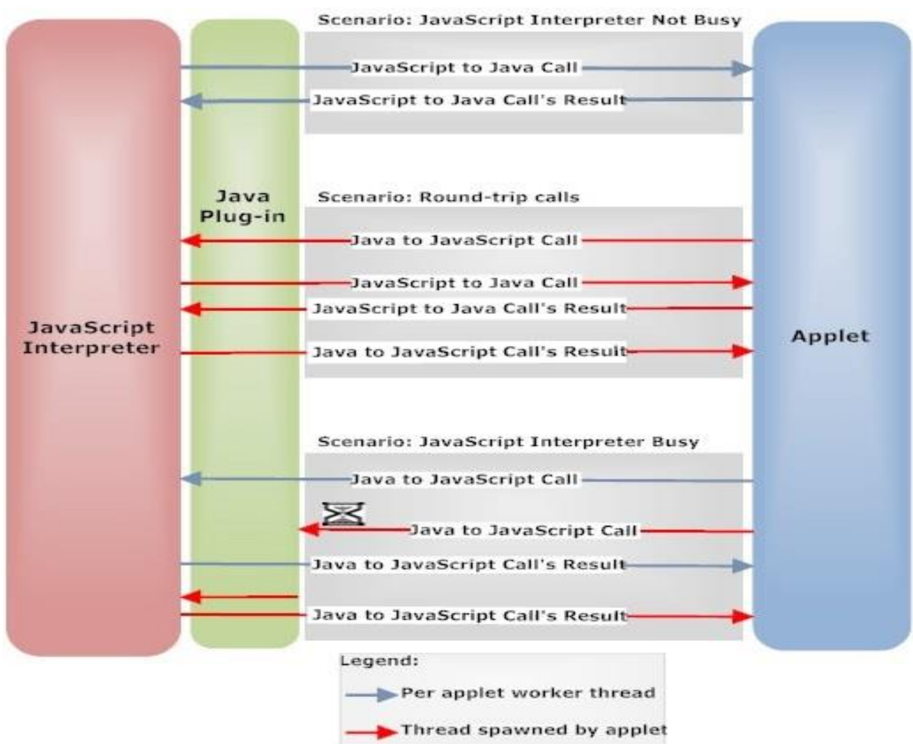


أدوات Java:

تأتي SDK المقدمة من شركة Sun مع مجموعة من الأدوات التي تمنح المبرمجين كل ما يحتاجونه لتطوير برامج Java. وتشتمل الأدوات المقدمة مع SDK 2 Java على أداة تجميع ومحول وأداة إنشاء للتعليمات وأداة لتصحيح الأخطاء، علاوة على مجموعة كبيرة من مكتبات البرامج المنشأة مسبقًا والتي تستخدم لإنشاء برامج Java. علاوة على ذلك، تزود SDK بأداة Viewer Applet الملائمة التي تسمح لك بتشغيل applets Java خارج متصفح الويب المتوافق مع Java. (بعض المتصفحات مثل HotJava وNavigator وNetscape Explorer (Internet تستهلك المصادر بشكل كبير وبالتالي تجعل عملية التطوير بطيئة للغاية. في حين تتسم Viewer Applet بصغرها وسرعتها وكفاءتها، مما يجعل من السهل اختبار البريمجيات Applet في مراحل مبكرة.

البُريمجيات Java applets:

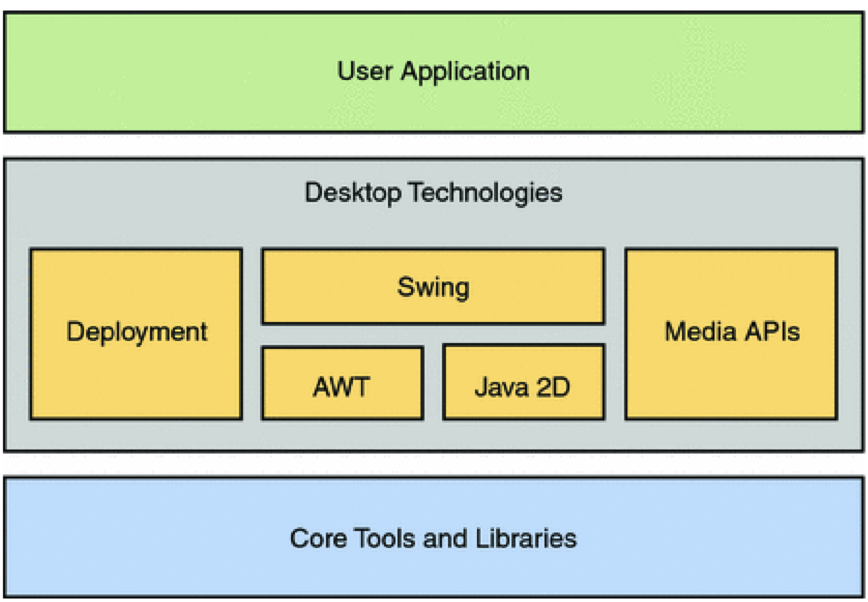
Java applets هـي عبـارة عـن بـرامج صـغيرة يمكـن إدخالهـا إلى صـفحات الويـب مباشـرة، وبالتـالي تحتـاج هـذه الـبرامج إلى متصـفح يتوافـق مـع لغـة Java (أي متصـفح يشـتمل بداخلـه علـى بيئـة تشـغيل Java حتـى يمكـن تشـغيلها) وتعتـبر هـذه الـبرامج أجـزاء مـن كـود Java التنفيـذي والـتي يـتم تضـمينها في مسـتندات HTML باسـتخدام علامـة الترميـز <APPLET>. يجـب إدخـال Java 2 applets في صـفحات الويـب باسـتخدام علامـتي الترميـز <OBJECT> و<EMBED>، وعنـدما يصـل أحـد المتصـفحات المعتمـدة علـى Java إلى مثـل هـذه الصـفحة، فإنـه يقـوم تلقائيًـا بتنزيـل الكـود التنفيـذي المشـار إليـه بعلامـة الترميـز الخاصـة. وعنـدما يصـل الكـود، تقـوم بيئـة تشـغيل Java بتنفيـذه داخل المتصفح.



تختلف البريمجيات Applets عن التطبيقات لأنه لا يمكن تشغيلها بذاتها، بل يجب تشغيلها داخل أحد المتصفحات المتوافقة مع Java أو باستخدام أداة خاصة مثل Applet Viewer.

تطبيقات Java:

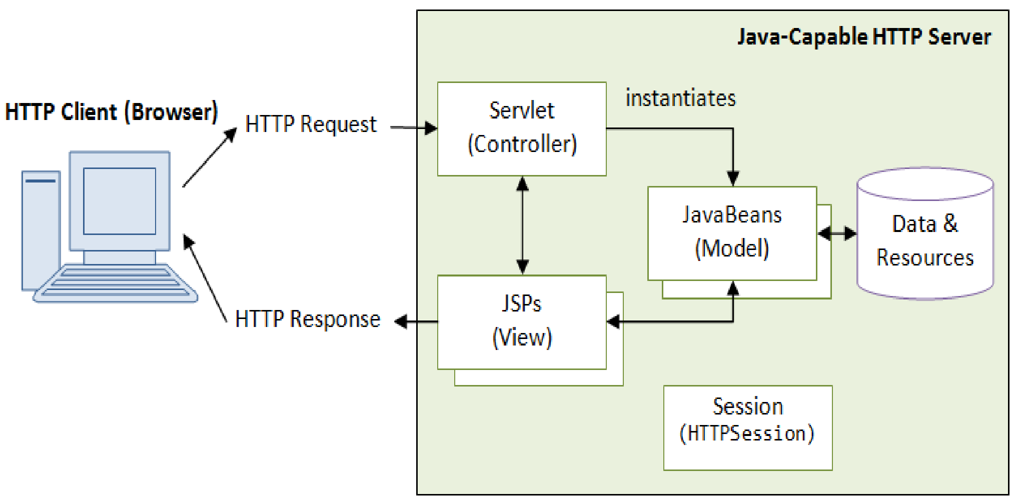
تعرف تطبيقات البرامج المستقلة المكتوبة باستخدام لغة Java باسم تطبيقات Java. يمكن تشغيل مثل هذه التطبيقات على سطح المكتب، خارج البيئة المتوافقة مع Java (مثل متصفح الويب المعتمد على Java أو أداة Applet Viewer). على سبيل المثال، يعتبر متصفح HotJava الذي قدمته شركة Sun أحد تطبيقات Java المستقلة بالفعل والذي يمكنك تشغيله على سطح مكتب الكمبيوتر مثلما تقوم بتشغيل أي برنامج آخر. وعلى الرغم من أن Java applets قد لاقت في البداية كل الانتباه لأن بإمكانها منح صفحات الويب الجامدة المحتوى التفاعلي، إلا أن تطبيقات Java قد حازت نصيبها من الشهرة بمرور الوقت.



برامج Java الأخرى:

على الرغم من أن البريميجيات applets والتطبيقات هي أشهر الأشكال التي توجد عليها برامج Java، إلا أنه يمكن إنشاء الأنواع الأخرى من برامج Java أيضًا، بما في ذلك JavaBeans والـServlets والمعالجات.

إن JavaBeans هي عبارة عن مكونات برامج Java التي يمكن تجميعها بشكل مرئي دون الحاجة لمبرمج محترف باستخدام أداة إنشاء مرئية. وعلى عكس فئات Java التي يجب أن يجمعها مبرمج محترف في البرامج الوظيفية، يتم تصميم JavaBeans خصيصًا لكي يتم التعامل معها بشكل مرئي من خلال واجهة مرئية مشابهة لتلك الخاصة بـBasic Visual وأدوات تجميع البرامج المرئية الأخرى. فمثل هذه الأدوات تسمح لغير المبرمجين «بربط» مكونات برامج JavaBeans معًا على مستوى مرئي، وتحريرها من التكامل الخاص بمستوى الكود الذي اعتدنا عليه باعتبارنا مبرمجين.



يمكننا اعتبار الـServlets على أنها applets خاصة بجانب وحدة الخدمة، أو على أنها برامج صُممت لغرض خاص تعمل على وحدة خدمة الويب بدلاً من متصفح المستخدم النهائي. تعتبر Java Servlets بديلاً لنصوص Common Gateway Interface (CGI) الشهيرة التي تكتب بلغة PERL. ويمكن التحكم في الـServlets من خلال أوامر البرمجة المضمنة في صفحات الويب الخاصة بجانب الوحدة التابعة بطريقة مشابهة لتلك الخاصة بـ(Active Server Pages (ASP التي قدمتها شركة Microsoft.

المعالجات هي أجزاء خاصة من كود Java تعالج المعلومات الواردة وتحولها إلى كائن يمكن لبيئة Java استخدامه. وعلى الرغم من استخدام المعالجات بصفة أساسية فقط في متصفح HotJava الذي تقدمه شركة Sun، فإن هذه المعالجات توضح كيف يمكن استخدام Java للتطوير الديناميكي لإمكانيات متصفحات الويب والبرامج الأخرى التي تواجه معلومات جديدة مجهولة بشكل متكرر. وفي الأساس، توفر المعالجات لمتصفح HotJava آلية تعلم كيفية التعامل مع البيانات الواردة.

تقوم Java بدعم نوعين من المعالجات: معالجات البروتوكول ومعالجات المحتوى. تعمل معالجات البروتوكول على زيادة معرفة HotJava بالبروتوكولات. فلو واجه المتصفح بروتوكول لا يفهمه، فإنه يستطيع معرفة هذا البروتوكول ديناميكيًا ببساطة عن طريق تنزيل وتنفيذ المعالج المرتبط بهذا البروتوكول، إذا وجد. في حين تعمل معالجات المحتوى على زيادة معرفة HotJava بالمحتوى، مثل الصور والأصوات والنماذج الأخرى من المعلومات المعتمدة على الويب.

توصيف Java:

والآن وبعد أن عرفت المزيد من المعلومات عن Java، نرى أنه قد حان الوقت لمعرفة ما هو أكثر من هذه المفاهيم أي التعرف على بيئة Java بأكملها - لغة Java وبيئة التشغيل والأدوات - بصورة أكثر تعمقًا. وأفضل الطرق لتحقيق ذلك التركيز على التقنية نفسها.

تتألف Java من ثلاثة كيانات:

- اللغة ذاتها.
- بيئة التشغيل.
- مجموعة من الأدوات.

تعتبر لغة البرمجة هي أساس كل شيء، وقد كانت تعرف سابقًا باسم Oak، ثم تم تغيير هذا الاسم إلى Java نظرًا لأسباب قانونية (فقد كانت هناك تقنية مختلفة تسمى Oak أيضًا قبل لغة Oak التي قدمتها شركة Sun). وبدون لغة Java لن يكون لبيئة التشغيل والأدوات أية فائدة. عند الحديث عن Java، عادة ما تشير الكلمة إلى لغة البرمجة.

ولكن ما الذي يجعل هذه اللغة تنال هذا القدر من الأهمية؟ ألم نر عدد من لغات البرمجة عبر السنوات فشلت بطريقة أو بأخرى في تحقيق هذا القدر الكبير من الشهرة؟ لقد اعتدنا في مجال تطوير البرامج على ظهور لغات جديدة تعد بتغيير الطريقة التي نبرمج بها وتحدث ثورة في تطوير البرامج وتسهل من وظائفنا وتعمل على تقدم حالة مجال التطوير بأكمله. فلماذا نؤمن بأهمية هذه اللغة إذًا ونحاول تعلم نموذج جديد من البرمجة؟

كل هذه الأسئلة وجيهة ومنطقية، وسوف نجيب على معظمها - بإذن الله - خلال دراستنا، في حين سيجاب على بقيتها في السنوات القادمة، بالنسبة للوقت الحالي لا تزال Java لغة جديدة نسبيًا. فعلى الرغم من كونها متاحة منذ عام ١٩٩٥، إلا أن استخدامها لم يتزايد سوى الآن لتثبت أنها تستحق الشهرة التي نالتها في بداية ظهورها على الساحة منذ تسع سنوات مضت. ظل الأشخاص الذين يعملون بمجال الكمبيوتر لسنوات يتحدثون عن إمكانيات Java، ومع الإصدار الثاني من Java، أصبحت التقنية جاهزة لتحقيق ما وعدت به في البداية. ولكن ماذا يجعل Java لغة خاصة؟ لاكتشاف الإجابة عن هذا السؤال. يمكننا البدء بالرسالة التسويقية المثيرة التي قدمتها شركة Sun عند ظهور Java 1.0 للمرة الأولى في صيف عام ١٩٩٥.

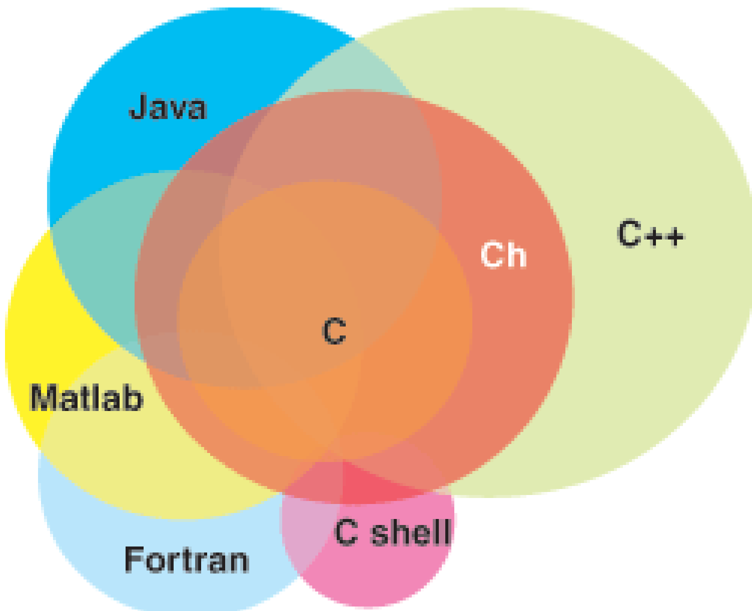
> Java: لغة بسيطة ولغة برمجة كائنية التوجه Object-Oriented موزعة ومفسّرة (محولة) وقوية وآمنة وتعمل على أي نظام تشغيل بغض النظر عن التطبيق، كما يمكنها تنفيذ عمليات متعددة وهي لغة ديناميكية ذات مستوى مرتفع من الأداء.

تعتبر كل صفة من هذه الصفات هدفًا في حد ذاتها في مجال تطوير البرامج الحديثة. ولذلك فإننا لن نسلم بهذا التعريف دون مناقشة، بل سنناقشه أولاً لنر كيف تجسم لغة Java هذه السمات والإمكانيات بالفعل.

لغة بسيطة:

بالنسبة للمبرمجين الخبراء تبدو لغة Java بسيطة عند مقارنتها بالكثير من لغات البرمجة الأخرى المعقدة. ولكنها ليست بسيطة للغاية، أنها بسيطة بدرجة كافية لمواصلة العمل بدون الاضطرار إلى قضاء وقت في تعلم لغة جديدة تمامًا. أما

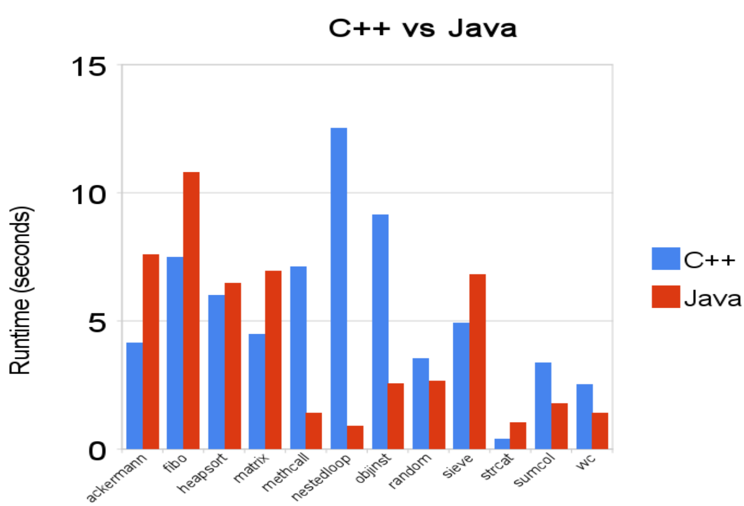
بالنسبة للمبتدئين، فقد تم تصميم Java لكي تكون قريبة من لغتي C و++C بقدر الإمكان، وذلك لضمان أن القاعدة العريضة من المبرمجين الذين يستخدمون C و++C هذه الأيام يمكنهم الانتقال إلى استخدام اللغة الجديدة بسهولة. ونظرًا لأن Java لها جذورها في C و++C، يمكن لأي شخص معتاد على التعامل مع هاتين اللغتين أن يعتاد سريعًا على استخدام Java في وقت لا يذكر.



على الرغم من ذلك، سوف يحتاج مبرمجو C لقليل من الوقت في الاعتياد على استخدام Java وذلك مقارنة بمبرمجي ++C. حيث أن Java عبارة عن لغة تدعم استخدام الكائنات وهي تشبه لغة ++C في ذلك بدرجة كبيرة، في حين أن لغة C تعتمد على الإجراءات وتشبه Java بصورة رئيسية في تركيبها. نتيجة لذلك، يجب على مبرمجي C أولاً أن يتعلموا التفكير في إطار الكائنات، في حين أن مبرمجي ++C معتادون على العمل باستخدام الكائنات.

لحسن الحظ، تعتبر Java أسهل في تعلمها من C++ لأن مصممي لغة Java قاموا متعمدين بإزالة غالبية الصعوبات التي واجهها المبرمجون مع لغة C++. وبالفعل، تم إنشاء Java بحيث تكون أسهل في تعلمها واستخدامها من C++، ونتيجة لذلك، تخلت عن بعض إمكانيات C++ القوية. إلا أن هذه الإمكانيات لا تستخدم بصورة متكررة كما قد تتصور، بل أنها غالبًا ما تؤدي إلى إنشاء برامج معقد صعبة التعديل.

وعلى وجه التحديد، تم حذف المؤشرات التي تشير لموضع في الذاكرة وتعيين أكثر من وظيفة لمعامل واحد واشتقاق فئة جديدة من العديد من الفئات الموجودة بالفعل وتحويل البيانات التلقائي من لغة Java (إلا أن التحميل الزائد للعملية مازال مدعمًا).

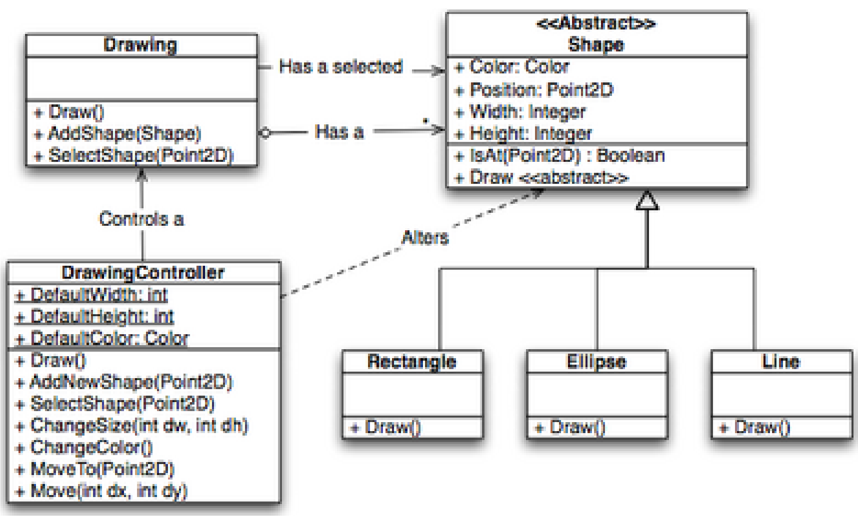


ربما كانت هذه الأشياء تفيد مبرمجي لغة C++ ولكنها كانت تعقد اللغة، ولذا فقد أزالت لغة Java هذه الإمكانيات لكي تظل بسيطة وسهلة الاستخدام. وتلك هي أول النقاط التي تحرزها في مجال البساطة.

وفي خلال عملية التبسيط قام فريق Java بتقديم طريقة للتجميع التلقائي للمحذوفات. وعلى الرغم من أن ذلك قد زاد من درجة تعقيد نظام Java الداخلي، فقد عمل على إزالة عبء إدارة الذاكرة عن المبرمج، وهكذا، لا يتسم برنامج Java بسهولة كتابته فحسب، بل إن المشكلات المرتبطة بالذاكرة قد قلت إلى حد كبير. وتلك هي ثاني النقاط التي تحرزها Java في مجال البساطة.

لغة برمجة كائنية التوجه Object-Oriented:

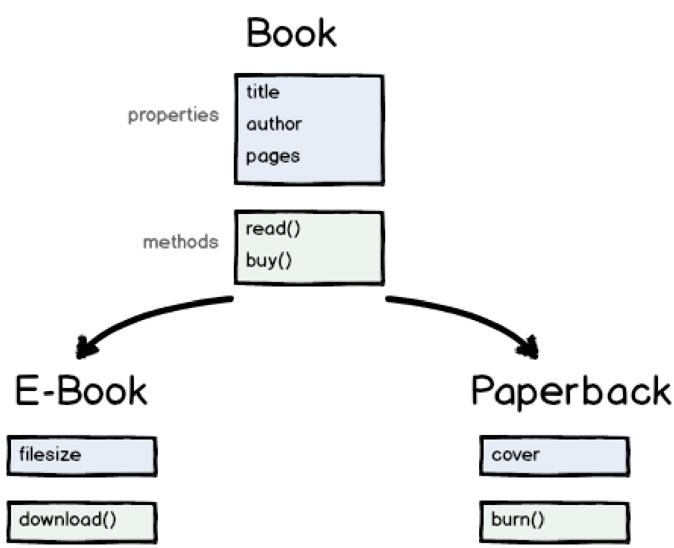
لو كانت لغة Java تتسم بالتعقيد، فإن ذلك لأن اللغة تدعم استخدام الكائنات بشكل كامل. فبالنسبة للمبرمجين غير المحترفين، يمكن أن يمثل مصطلح البرمجة المعتمدة على الكائنات (OOP) مفهومًا مروعًا، أما بالنسبة لهؤلاء المعتادين على التعامل مع الكائنات، فإن ذلك الأمر سهل للغاية.



يتسم منطق البرمجة التي تدعم استخدام الكائنات بالبساطة: فكل مهمة برمجة ينظر إليها في إطار الكائنات وعلاقاتها ببعضها البعض. ومن ناحية عامة،

إن الكائن هو عبارة عن وحدة مستقلة من الكود القابل للتنفيذ، تحتوي الكائنات على بيانات وطرق أو وظائف تتعامل مع البيانات، وعلى العكس من ذلك، تتعامل اللغات التي تعتمد على الإجراءات، مثل C، مع البيانات والوظائف (التوابع) على نحو منفصل، حيث يتم نقل البيانات إلى تابع، فيتعامل معها ويرجعها إلى المستدعي.

ومن ناحية أخرى، تتشابه الكائنات في طبيعتها مع الخلايا الحيوية التي تتألف منها أجسامنا. فمثل الخلايا، تحتوي الكائنات على معلومات بداخلها وهي تؤدي مهمتها من خلال هذه المعلومات. إن الكائنات هي وحدات مستقلة يمكنها العمل بشكل مستقل أو التجمع لإنشاء هيكل أكثر تعقيدًا. وبالمثل، كل خلية في جسمنا عبارة عن كيان مستقل منفرد، وتقوم هذه الخلية بوظيفة خاصة ولكنها قد تتجمع لإنشاء تركيبات أكثر تعقيدًا كالأنسجة. والنسيج، بدوره، يمكن جمعه مع الأنسجة الأخرى لتكوين أعضاء كاملة، كما يمكن جمع الأعضاء لتكوين أجهزة بأكملها مثل الجهاز التنفسي والجهاز الهضمي.

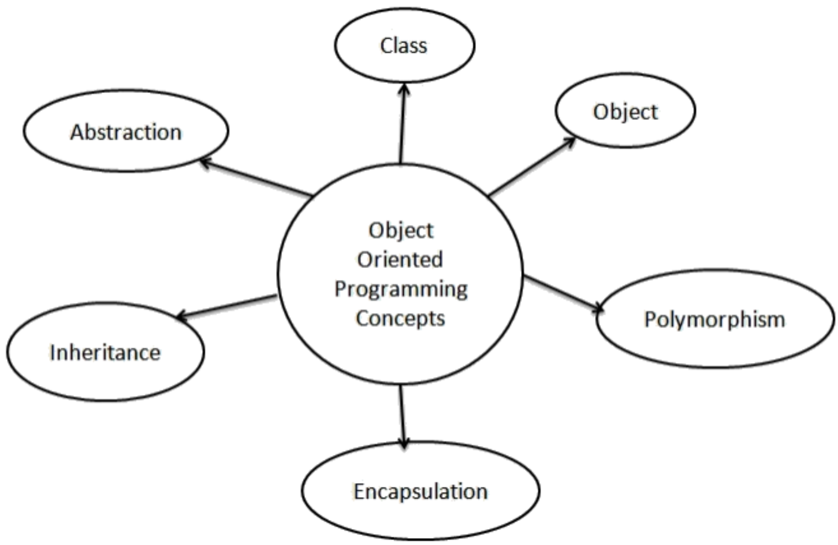


في الأصل، تتبع الكائنات القاعدة ذاتها التي تتبعها الخلايا، فالكائنات هي وحدات مستقلة يمكن جمعها مع كائنات أخرى لإنشاء تركيبات أكثر تعقيدًا وهي البرامج. وأحد الصفات التي تدعمها الكائنات هي الاشتقاق. فالعناصر الجديدة يمكن إنشاؤها، أو اشتقاقها، من العناصر الموجودة. ويكتسب العنصر الجديد إمكانيات العنصر الرئيسي، ولكن من الممكن إكسابه صفات جديدة حسب الحاجة. ويمنح الاشتقاق اللغات التي تدعم استخدام الكائنات مثل Java آلية رائعة لإعادة استخدام الكود، مما يعني أنك تستطيع عمل الكثير من الأشياء في وقت قليل.

تلميح: **الفئة**: هي كود أساسي متكامل يعمل بمثابة قالب تُنشأ منه الكائنات. وباعتبارك مبرمجًا، سوف تقوم بكتابة كود Java لإنشاء الفئات التي يبدأ عملها عند تشغيل البرنامج بوصفها كائنات. وفي إطار الصورة السابقة، يمكن مقارنة الفئات بتركيب DNA للخلية، في حين سوف تمثل الكائنات خلية حية فعلية. كما يمكن مقارنة فئات Java بالتصميم الهندسي لمنزل، في حين يمثل الكائن المنزل المنشأ من هذه التصميمات. وعلى الرغم من أن هذه النظرة مبسطة للغاية حول العلاقة بين الفئات والكائنات، فإنها سوف تكفي في الوقت الحالي.

ولكي نوضح القيمة العظيمة للبرمجة التي تدعم استخدام الكائنات بشكل ملموس فلنتصور أنه مهمة كتابة البرنامج تتماثل مع إنشاء تمثال باستخدام كتل البناء. فستفرض عليك لغات البرمجة التي تدعم الإجراءات عمل كل كتلة بناء

يدويًا من البداية، الأمر الذي يحتاج إلى قدر كبير من الوقت والجهد من جانب المبرمج، ويجب الانتباه إلى شكل كل كتلة بناء وتناسبها داخل العمل.

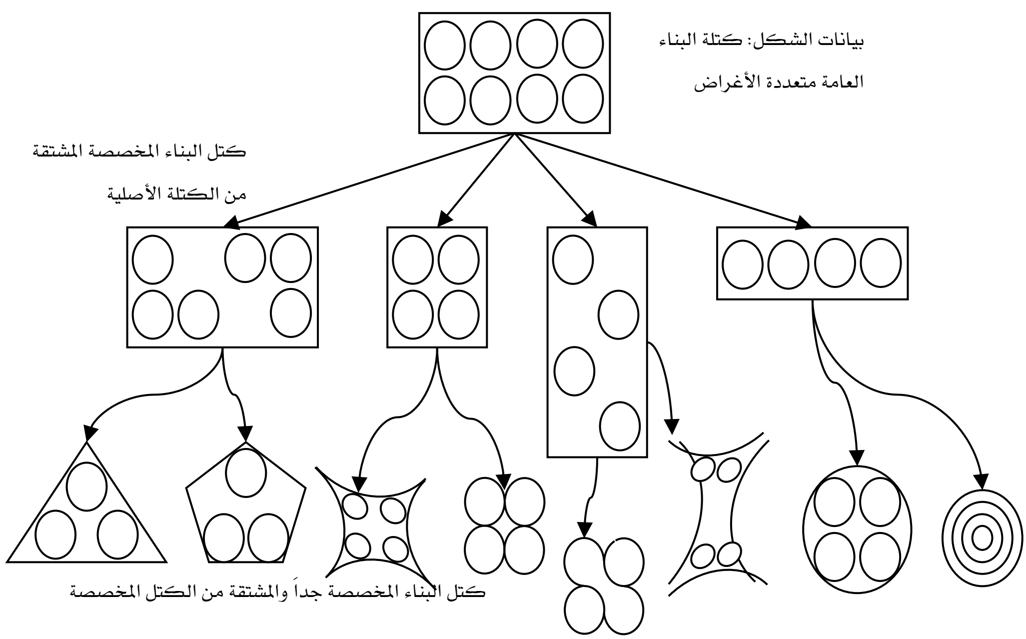


أما البرمجة التي تدعم استخدام الكائنات فهي تركز على العمل الأكبر ذاته، على الشكل النهائي الذي ينبغي أن يكون عليه التمثال. وبدلاً من إنشاء كتلة بناء يدويًا لكل جزء من الكل، فهي تعتمد على نموذج عام من كتل البناء يتم منه اشتقاق كل كتل البناء الأخرى. وسوف توفر الوقت والجهد المهدرين في إنشاء كتل البناء كل على حدة، فكل ما عليك أن تفعله هو أن تستخدم كتلة البناء العامة كقالب وتنشئ بناء عليها. يعطيك القالب كل السمات الأساسية، وعليك أن تعدله بحيث يتناسب مع احتياجاتك. وستكون النتيجة هي زيادة الإنتاجية وإعادة الاستخدام بشكل رائع.

في مجال البرمجة كائنية التوجه Object-Oriented يكون التركيز بصورة رئيسية على تصميم كتل البناء (الكائنات) وعلى العمليات التي تتم على هذا

التصميم. ولو أنك قمت على نحو جيد بتوصيف كتلة البناء الرئيسية متعددة الأغراض، يمكنك بعد ذلك اشتقاق مجموعة كبيرة من كتل البناء ذات الأغراض الخاصة منها.

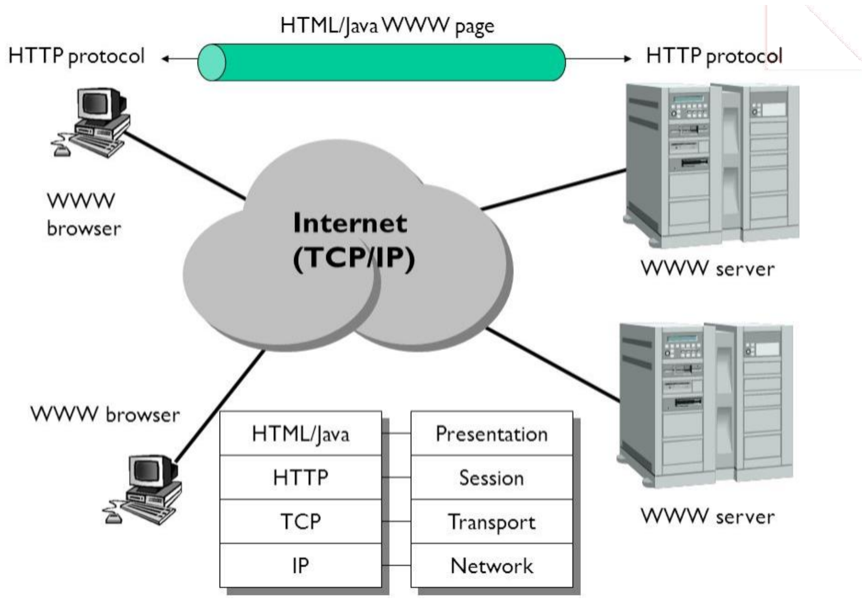
إن الوظائف التي تعتمد على الكائنات في لغة Java هي نفسها تلك الخاصة بلغة C++. وعلى الرغم من أن مبرمجي C++ الحاليين لن يواجهوا أي متاعب عند استخدام Java ، فإن المبرمجين الذين ليست لديهم خبرة بالتعامل مع الكائنات سوف يلزمهم بعض الوقت للاعتياد على هذه التقنية والمفاهيم المرتبطة بها.



لغة موزعة:

تم إنشاء لغة Java مع وضع اتصالات الشبكة في الذهن، فالإمكانيات المعقدة لشبكة الاتصالات هي جزء جوهري في تقنية Java، تشتمل Java على مكتبة شاملة من كائنات الشبكة التي تتعامل من البروتوكولات مثل: Transmission

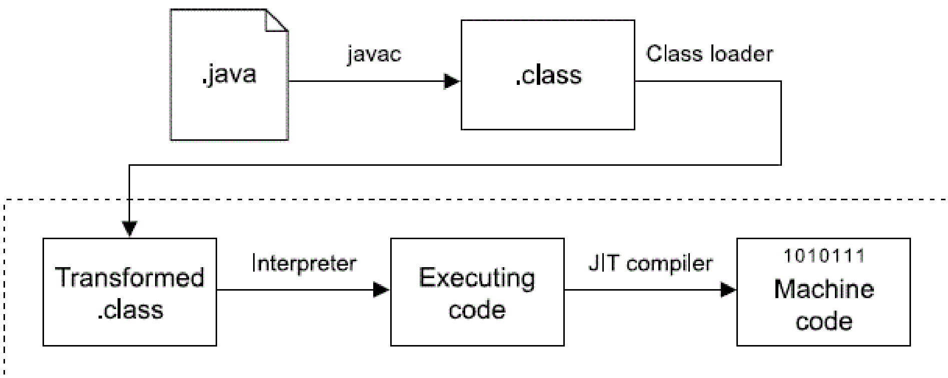
Hypertext Transfer Protocol و Control Protocol/Internet Protocol (TCP/IP) (HTTP) و(FTP) File Transfer Protocol. نتيجة لذلك، يمكن لبرامج Java فتح المصادر الموجودة على الإنترنت والوصول إليها بنفس سهولة الوصول إلى نظام ملفات محلي.



وفيما يلي ميزة أخرى قوية تتميز بها Java فليس عليك أن تنفذ تفاصيل الاتصال الشبكي بنفسك، فلغة Java توفر كل ما يلزم لعمليات المعالجة الموزعة. ومثال جيد على ذلك هو متصفح HotJava الذي بفضل لغة Java، يفهم كيفية الوصول إلى والتعامل مع الكائنات عبر الويب. تتطلب منك غالبية لغات البرمجة الأخرى أن تكتب مستويات الشبكة هذه بنفسك، أما Java فتقدم هذه الإمكانيات من البداية.

لغة مفسّرة (محوّلة):

إن لغة Java هي لغة مفسّرة (محولة) (interpreted Language). بعبارة أخرى، يجب إتاحة مفسر لكل بنية جهاز ونظام تشغيل ترغب في تشغيل تطبيقات Java عليه. قديمًا، كانت البرامج المفسرة لا تحول إلى كود الجهاز، كما يحدث في البرامج المجمعة (المترجمة). وبدلاً من ذلك، كانت هذه البرامج تبقى بشكل يمكن للإنسان قراءته وكذلك ينفذه المفسر سطرًا بعد سطر.



غالبًا ما ينتاب مطورو البرامج الخوف عند تفكيرهم في استخدام لغة مفسرة لمشروعات البرمجة «الكبيرة». ومن المعروف أن اللغات المفسرة بطيئة وذلك مقارنة بشبيهاتها المجمعة (المترجمة) وغالبًا ما يوجه إليه النقد لهذا السبب.

على الرغم من ذلك، ومن خلال الاستخدام المبتكر لـ Java virtual machine (JVM)، تغلبت Java على الكثير من مشكلات السرعة التي كانت سببًا في فشل اللغات المفسرة الأخرى. ويمر كود Java خلال عملية تجميع تنتج ما يعرف باسم بايت كود Bytecode. والبايت كود وهو مطابق لمواصفات Java Virtual Machine، ويمكنك تشغيله على أي نظام يدعم بيئة Java. يكون هذا الكود في مرحلة

متوسطة، أي أنه ليس مجمعًا بصورة كاملة، ولكنه قريب بدرجة كافية لكود الجهاز الفعلي الذي تستغرق فيه عملية التحويل وقتًا أقل مما يحدث مع الأكواد الأخرى.

لتقليل عبء التحويل بدرجة أكبر، تطبق Java تعدد العمليات (الذي يسمح للبرامج بالتعامل مع العديد من المهام في وقت واحد) وتتعامل مع التحويل بفاعلية ويؤدي تجميع كود Java في مرحلة متوسطة - ثم ترجمته في الخلفية من خلال العمليات - إلى تفسير أسرع كما أنه يعطي Java سرعة كبيرة تميزها عن اللغات المفسّرة التقليدية.

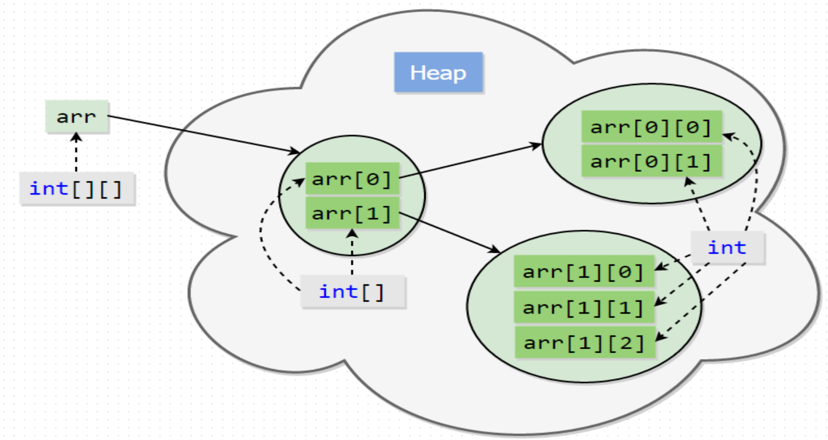
ملحوظة: إن Java Virtual Machine (JVM) ليس جهازًا، فهو مجموعة من المواصفات التي تحدد كيف يجب التعامل مع البايت كود Bytecode الخاص بـJava على نظام كمبيوتر محدد. ويمكن لبرامج Java أن تعمل على أي كمبيوتر به Java Virtual Machine.

لغة قوية:

تعتبر الثقة شيئًا مهمًا للغاية بالنسبة للمبرمجين، وخاصة المحترفين الذين تعتمد سمعتهم ووظائفهم على المنتجات التي ينشئونها. لذلك، يجدر بنا إنشاء برامج قوية محتمل لها النجاح في ظل ظروف غير معروفة أو ظروف مغايرة. وعلى الرغم من أن هذا الهدف ربما يبدو واضحًا، فإن الكثير من اللغات هي أبعد ما تكون عن توفير الأدوات اللازمة للمبرمجين لتطوير برامج قوية.

تحتاج لغة Java إلى توصيفات تضمن أن أنواع البيانات التي يتم تمريرها إلى طريقة (تابع) ما تكون هي بالفعل البيانات التي تحتاجها هذه الطريقة. علاوة على

ذلك، لا تسمح Java بالتحويل التلقائي لنوع البيانات، كما تسمح به كل من C وC++، إذ يجب على المبرمج كتابة التحويلات بوضوح. ونتيجة لذلك يقل احتمال ظهور الأخطاء في الكود.

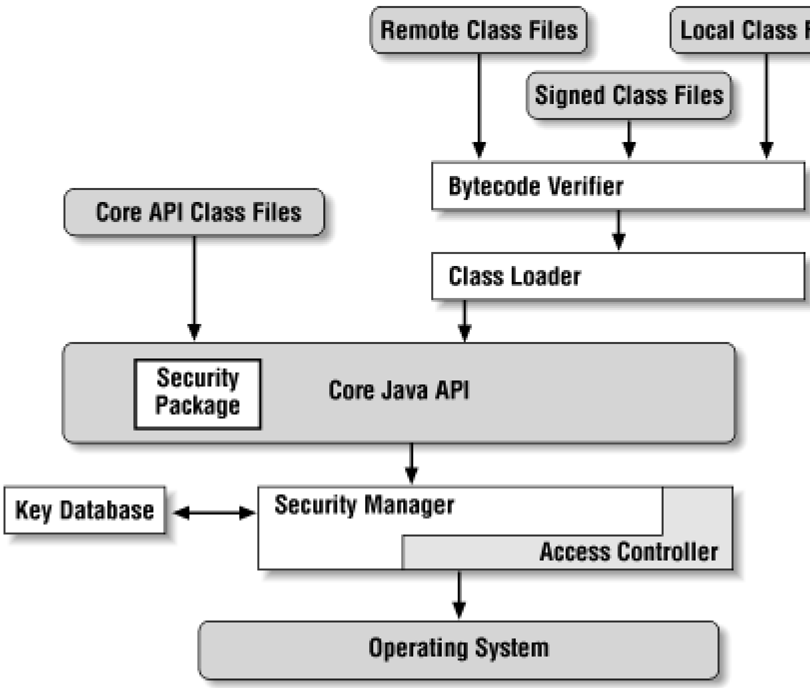


يتمثل أكثر الاختلافات وضوحًا بين Java وC/C++ في استخدام متغير يحتوي على موقع البيانات في الذاكرة يمنع حذف الذاكرة وإتلاف البيانات. وبدلاً من حساب هذا المتغير، تقوم Java بدعم المصفوفات الفعلية التي تدعم تحديد المتغيرات التي تحدد موقع عنصر في المصفوفة (الحدود). ونظرًا لعدم قدرة المبرمج على حذف البيانات أو الوصول إلى عنصر مصفوفة خارج الحدود على سبيل الخطأ، فإن هذه الأخطاء لا تحدث في لغة Java. وتتمثل النتيجة في نظام تطوير قوي جدًا يحرر المبرمج من مثل هذه الأعباء.

لغة آمنة:

في أي وقت يحدث فيه اتصال بالشبكة، تصبح الاعتبارات الأمنية مسألة خطيرة ومهمة. لقد كانت مسألة التأمين مبعث قلق دائم، حتى على المستوى

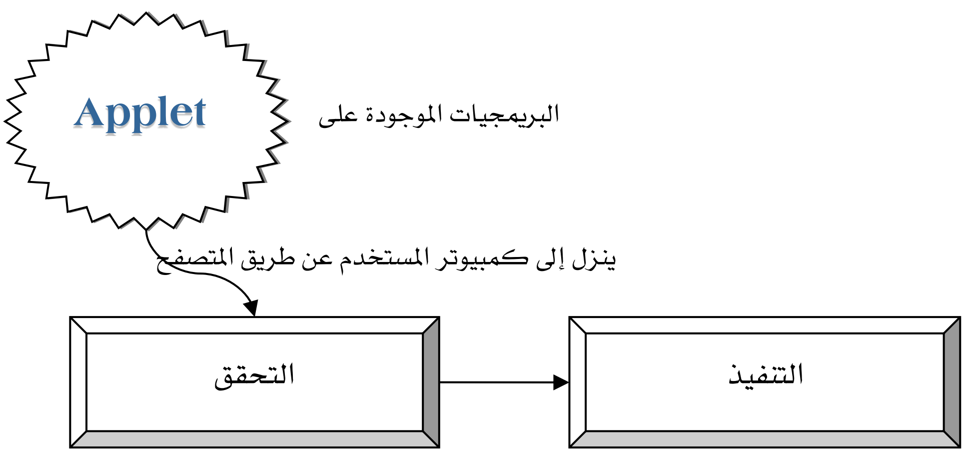
المحلي وقبل أن تحقق شبكات الاتصالات هذا القدر من الانتشار والشهرة. والآن وبعد وجود المحتوى التنفيذي، زادت الفيروسات والمتطفلون الذين يأتون في صورة متخفية ليقتحموا سرية المستخدمين. حيث لا توجد وسيلة خير من التخفي للدخول إلى نظام الكمبيوتر الخاص بأحد المستخدمين. ونظرًا لأنه يتم تنزيل الـapplets وتشغيلها تلقائيًا، فلن يمكن معرفة سبب المشكلة أو المكان الذي أتت منه.



من البداية، قام فريق Java بتصميم لغة Java من أجل البيئات الموزعة والمرتبطة فيما بينها بواسطة شبكات اتصالات، وقد تعامل مع مشكلة التأمين مبكرًا. يعتبر نظام التأمين جزءًا مهمًا من بيئة Java. لذلك، فهي تمكنك من إنشاء نظم خالية من الفيروسات والأخطاء، وذلك بفضل عدد عمليات التحقق التي تقوم بها هذه اللغة قبل تنفيذ أي جزء من الكود.

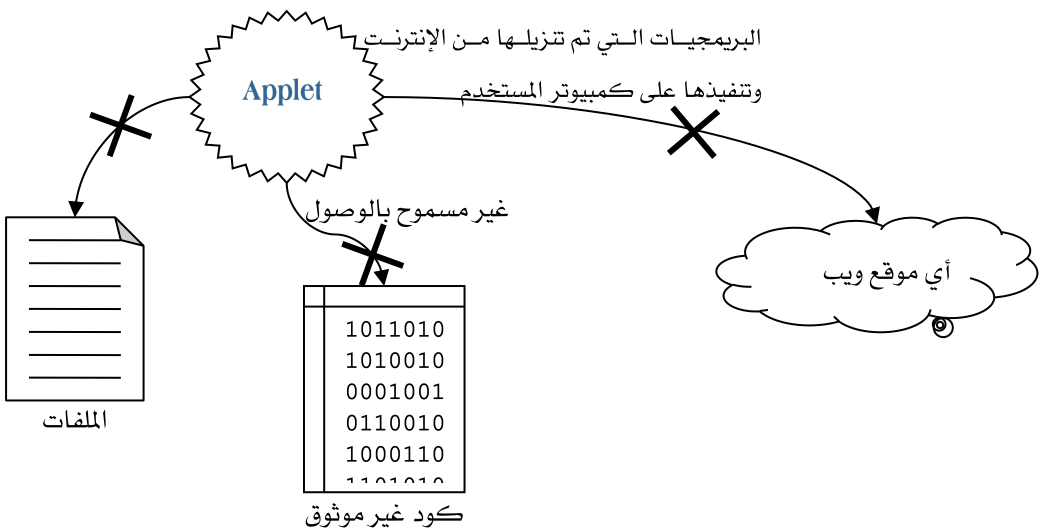
أولاً، يتم إزالة كل من المؤشرات وتخصيص الذاكرة أثناء وقت التجميع. وعن طريق منع تخصيص الذاكرة والوصول إلى الذاكرة أثناء وقت التشغيل، تمنع Java المبرمجين من افتراض مواضع الذاكرة مسبقاً والوصول إلى الذاكرة.

ثانيًا، يتحقق المفسّر من كل الأكواد قبل أن يتم تنفيذها. فلو لم يكن الكود أحد الأكواد الصحيحة الخاصة بـJava فلن يتم تنفيذه، ويتم منع الكود غير القياسي قبل أن يمثل مشكلة. علاوة على ذلك، لا يسمح بالأنشطة الخطيرة مثل الوصول غير المشروع لمساحة الذاكرة وخرق امتيازات الوصول والوصول غير المشروع للفئات وعمليات تحويل البيانات غير المشروعة. نتيجة لهذا، لا يمكن لأي فيروس الوصول إلى هياكل البيانات أو الكائنات أو مواضع الذاكرة.



وفي النهاية، تعمل كل الـapplets داخل نموذج أمني Java: إذ يتم التعامل معها على أنها كود غير موثوق به ينفذ في بيئة موثوق بها. يعني هذا أن كل الـapplets التي تنزل من الويب يتم تحديد صلاحياتها عندما تصل إلى جهازك. وحتى بعد تجاوز عملية التحقق من الكود، يتم منع الـapplets من الوصول للملفات الموجودة على جهاز الكمبيوتر الخاص بك إلا إذا قمت بصفة محددة بمنح الـapplet تصريح

لعمل ذلك. يعمل هذا على منع الـapplets من الوصول لمصادر الويب باستثناء تلك المصادر المتاحة على الموقع الذي تم تنزيلها منه، إلا إذا قام المستخدم بمنح الـapplet مثل هذه الإمكانيات. وكأحد الإجراءات التأمينية الإضافية، يتم منع الـapplets من تشغيل أي كود (مثل التطبيقات ومكتبات الكود الخارجية) لا يمكن التحقق منه بالطريقة ذاتها التي تم استخدامها في التحقق من الـapplet ذاته.



وعلى الرغم من استحالة اختراق النظام التأميني الذي وضعته Java من الناحية النظرية، إلا إنه في الحقيقة، يمكن اختراق أي نظام تأمين لأي نظام برمجي، بما في ذلك Java. وفي الحالات النادرة التي تم فيها اكتشاف ثغرات أمنية في نظام تأمين Java، قام فريق عمل Java الخاص بشركة Sun بسد هذه الثغرات على الفور. ولكن عند مقارنة Java بلغات البرمجة الأخرى، نجد أنها أكثر أمانًا ولذلك فقد أصبحت اللغة الفعلية المستخدمة لإنشاء برامج شبكات الاتصالات.

مستويات الصلاحيات:

تم تصميم applets x 1.0 Java لكي تعمل دائمًا داخل نموذج التأمين، مما يعني أنها لا تخرج أبدًا عن النطاق الضيق للصلاحيات التي تفرضها هذه النسخة. بعد ذلك، قدمت Java 1.1 نموذج تأمين مختلف، يمكن للمستخدمين من خلاله منح الـapplets وصول كامل لمصادر الشبكة والنظام أو حرمانها من هذه الصلاحيات تمامًا. ولسوء الحظ، سمح هذا النموذج التأميني الذي قدمته Java 1.1 بوصول الـapplets الكامل لكل المصادر حتى لو كان بحاجة إلى نوع واحد فقط من الوصول. ومع Java 1.1 لا يستطيع المستخدم تحديد أن يقرأ الـapplets الملفات على النظام المحلي فقط مثلاً، حيث أن منح الـapplets في Java 1.1 أي شكل من امتيازات الوصول يمنحه وصولاً كاملاً للمصادر المحلية ومصادر الشبكة بغض النظر عن الإمكانيات التي يحتاج إليها هذا الـapplets فعلاً.

قدم الإصدار الثاني من Java أخيرًا مفهوم «**مستويات الصلاحيات**» وهو نموذج تأمين يمكن منح الـapplets من خلاله إمكانيات محددة بناء على احتياجاته ومهامه. وعلى سبيل المثال، فإن الـapplets الذي يحتاج قراءة ملف موجود على كمبيوتر المستخدم يمكن منحه هذا الامتياز فقط ولا شيء غيره.

لغة تستطيع العمل على أي نظام تشغيل:

نظرًا لأن شركة Sun قامت بتصميم Java لدعم التطبيقات الموزعة عبر شبكات اتصال الكمبيوتر، يمكنك استخدام هذه اللغة مع مجموعة متنوعة من نظم التشغيل ووحدات المعالجة المركزية، ولتحقيق هذا الهدف، يقوم برنامج تجميع Java بإنتاج ملفات الكائنات التي تعمل على أي نظام تشغيل أو البايت كود

الخاص بلغة Java، من كود Java الأصلي الذي تقوم بكتابته. فمع افتراض أن بيئة تشغيل Java توجد على جانب نظام الوحدة التابعة، فإن الكود الناتج يتم تنفيذه بغض النظر عن النظام.

مع Java يمكنك إنشاء ملف تنفيذي واحد يمكن تشغيله على أي كمبيوتر يشتمل على بيئة تشغيل Java.

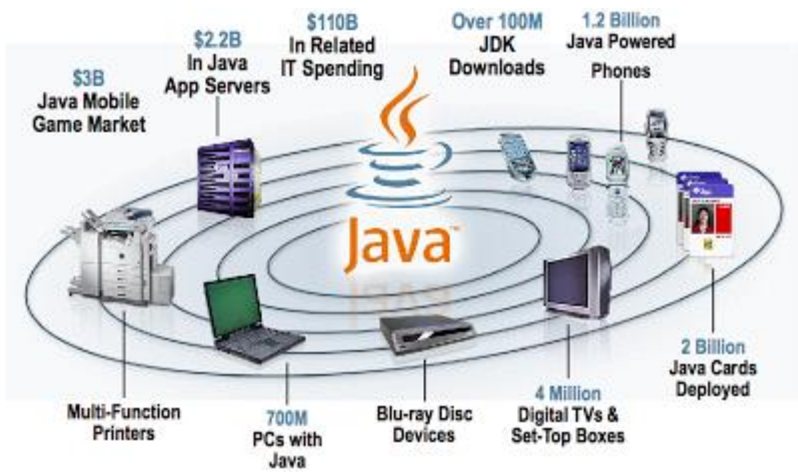
بيئة تشغيل Java:

إن بيئة تشغيل Java هي عبارة عن محول يقوم بتحويل ملفات البايت كود الخاص بـJava التي تعمل على أي نظام تشغيل إلى كود محلي خاص بالجهاز، وبالتالي يمكن تنفيذه. يتم تثبيت بيئة تشغيل Java على نظم المستخدم النهائي عندما يقوم المستخدم بتثبيت متصفح ويب متوافق مع Java مثل Netscape Navigator أو Internet Explorer. وعلى الرغم من أن هذه هي أشهر الطرق للحصول على بيئة تشغيل Java والعمل بها، فهي لا تفيد سوى في تشغيل Java applets المصممة في صفحات الويب. ولتشغيل تطبيقات Java كلها، يجب أن تثبت بيئة التشغيل الخاصة بـJava التي تعمل بصورة مستقلة عن متصفحات الويب. يكفي ذلك، لأن Java SDK الذي تقدمه شركة Sun وغالبية أدوات تطوير Java تأتي مع بيئة تشغيل Java المستقلة. نظرًا لأن بيئة تشغيل Java ما هي إلى أداة تحويل للبايت كود الخاص بـJava تتوافق مع JVM، فإن الأشخاص المسئولون عن إنشاء بيئات التشغيل لديهم مطلق الحرية في إنتاج مثلها. (غالبًا ما يقوم الناس باستخدام مصطلحي بيئة تشغيل Java وJava Virtual Machine ليعبروا عن الشيء ذاته لأن بيئة تشغيل Java تطبق بناء على مواصفات JVM في واقع الأمر، إنها مسألة وقت قبل أن يقوم كل المسئولين عن إنشاء نظم التشغيل بتجهيز بيئة تشغيل Java داخل نظام

التشغيل مباشرة. (تقوم كل من Microsoft وapple بإرفاق الإصدارات السابقة من Java داخل نظام التشغيل الخاصة بهما، ولكن الإصدار الثاني من Java لم يرفق بعد) وعندما تفعل الشركتان ذلك، فلن يضطر المستخدمون لتثبيت بيئة التشغيل بأنفسهم فسوف يتم دعم Java على مستوى نظام التشغيل.

تتاح هذه السمة الخاصة بالعمل على أي نظام تشغيل بفضل برنامج تجميع يقوم بإنشاء تعليمات الكود التي تعمل مستقلة عن أي نظام تشغيل. نتيجة لذلك، يمكن تشغيل برامج Java على أي كمبيوتر لأن الأكواد التي تتألف منها هذه البرامج يمكن أن تحولها بيئة تشغيل Java إلى كود محلي خاص بالجهاز.

بافتراض أن بيئة التشغيل الخاصة بـJava موجودة على كمبيوتر محدد، فإن البايت كود الخاص بـJava سينفذ بصورة صحيحة. وبفضل الإمكانيات السابقة التي ذكرناها، فإن نظام Java يمكن أن يعمل على أي نظام تشغيل بغض النظر عن التطبيق.



إن من أهم مميزات إمكانية العمل على أكثر من نظام بالنسبة لـJava تمثيلها لأنواع البيانات. على الرغم من اتسام كل من لغتي C وC++ بأنها يمكن العمل بهما على أي نظام، إلا نهما يتأثران بالتطبيق. يعني هذا أن حجم أنواع البيانات الأولية والعمليات التي تجرى عليها يتغير من نظام إلى آخر. فعلى سبيل المثال، نجد أن int (نوع بيانات الأعداد الصحيحة) على أحد الأنظمة لا يكون بالضرورة في مثل حجم int على نظام آخر. وقد يتسبب ذلك في مشاكل عند العمل على أنظمة مختلفة.

أما في لغة Java فيتم دائمًا تحديد حجم نوع البيانات الأولية وأداء العمليات التي تجرى عليها. على سبيل المثال، دائمًا ما يكون int عبارة عن قيمة عدد صحيح تتكون من ٣٢ بت وذلك تماشيًا مع مواصفات IEEE 754 الخاصة بـ Institute of Electrical and Electronics Engineers. وعندما تستخدم أنواع البيانات الأولية في Java ، فإن تمثيل هذه الأنواع والعمليات التي تجرى عليها تكون ثابتة على الرغم من اختلاف الأنظمة.

تشتمل مكتبات الفئات الخاصة بـJava على واجهات لكل نظام تشغيل تتاح عليه بيئة تشغيل Java. على سبيل المثال، نجد أن فئة Window التخيلية وعمليات التنفيذ الخاصة بها تكون متاحة لكل من Sun Solaris وWindows وMacOS. وعند استخدام مثل هذه الواجهات الرسومية، تقوم Java تلقائيًا بتحميل عنصر واجهة الاستخدام الرسومية الملائم الخاص بنظام التشغيل، ذلك العنصر الذي يقابل نافذة للجهاز الحالي.

في النهاية يتضح أن نظام Java بأكمله يمكن العمل به على أي نظام بغض النظر عن التطبيق.

مستوى مرتفع من الأداء:

يعتبر أداء Java ممتازًا بالنسبة للغة محولة، وذلك بسبب تطوير برامج تجميع (JIT) just-in-time (برامج التجميع الفورية) التي تقوم بتطوير البايت كود دون التأثير على العمليات الأخرى. وطبقًا لشركة Sun، يتم تشغيل برامج Java بنفس سرعة برامج C/C++ تقريبًا.

وعلى الرغم من أن تحويل البايت كود أسرع من تحويل لغات البرمجة الأصلية، نظرًا لأنه في مرحلة وسطى، إلا أن تصميم بنية Java يؤدي إلى تقليل تدفق البيانات أثناء وقت التشغيل. علاوة على ذلك، يتم تنفيذ عمليات متعددة في نفس الوقت، مما يؤدي إلى تحسين السرعة الكلية لملفات Java التنفيذية ملفات.exe.

ملحوظة: على الرغم من كون مستوى أداء Java ممتازًا عند مقارنتها باللغات المفسّرة الأخرى، إلا أن المبرمجين في الماضي قد أقروا بأنها بطيئة جدًا لدرجة تمنع استخدامها في الحالات التي تمثل فيها السرعة عاملاً مهمًا. وقررت بعض الإحصائيات أن برامج Java تعمل بصورة أبطأ من برامج C/C++ بحوالي ١٠ إلى ٣٠ مرة. وفي حين كان هناك شك في ما إذا كانت النسخة الأصلية من Java سريعة بدرجة كافية للتطبيقات المهمة، قامت شركة Sun بتطوير السرعة الكلية لـJava من خلال الإصدار الثاني من Java. علاوة على ذلك، قامت Sun مؤخرًا بإصدار تقنية HotSpot كجزء من نسخة 1.3 Java التنفيذية، الأمر الذي أدى إلى تصغير فجوة الأداء بين Java والكود الداخلي.

بالإضافة إلى ذلك، قامت العديد من شركات إنشاء أدوات البرامج الشهيرة، مثل IBM وSymantec وMicrosoft وMetrowerks، بتطوير برامج تجميع JIT التي تحول البايت كود الخاص بـ applet Java إلى كود داخلي مطور للجهاز فورًا، حيث يتم تنزيل هذا الكود عبر الويب قبل تنفيذه مباشرة. ونتيجة لظهور برامج تجميع JIT، قامت شركات إنشاء المتصفحات بزيادة سرعة تشغيل applet Java في منتجاتهم. وقد استفادت الإصدارات القليلة السابقة من كل من Netscape Navigator وExplorer Internet من برنامج تجميع JIT، مما جعل المتصفحات المتوافقة مع Java والتي لا تستفيد من تكنولوجيا JIT تبدو أبطأ عند مقارنتها بهذه المتصفحات.

لغة تستطيع تنفيذ عمليات متعددة في نفس الوقت:

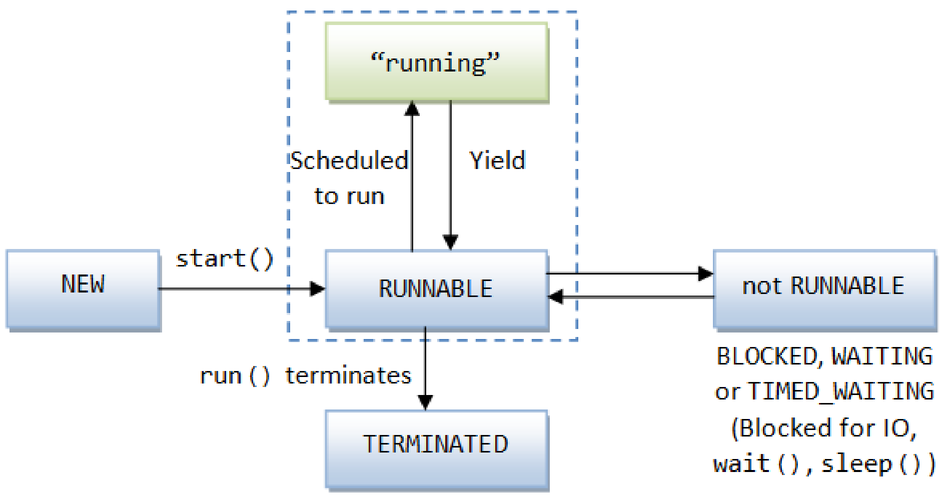
يعتبر تنفيذ عمليات متعددة في نفس الوقت إحدى الإمكانيات المهمة في Java، حيث أنها تتيح للملفات exe. القدرة على دعم تنفيذ العديد من العمليات في وقت واحد. ويجب ألا تخلط بين هذه الإمكانية وبين تعدد المهام، فتعدد المهام هو قدرة نظام التشغيل على تشغيل أكثر من برنامج في الوقت ذاته، في حين أن تعدد العمليات يسمح للبرنامج بتنفيذ العديد من المهام في وقت واحد.

على سبيل المثال في البيئة التي لا تتمتع بإمكانية تعدد العمليات، لا يمكن للبرنامج تنفيذ أكثر من عملية واحدة. حيث تستخدم الجملة التكرارية للحدث لتتبع المهمة المحددة الحالية. وتحدد هذه الجملة المهمة التي يجب تنفيذها (مثل تنشيط محتويات نافذة أو تتبع مدخلات المستخدم أو طباعة مستند) ثم تنقل هذه المهمة إلى الإجراء (الإجراءات) المناسبة. وبمجرد أن تنفذ المهمة، يمكن أن تبدأ

مهمة أخرى. ونظرًا لأنه لا يمكن التعامل سوى مع مهمة واحدة فقط، يجب أن ينتظر النظام حتى تنتهي المهمة الحالية قبل التعامل مع مهمة جديدة.

على الرغم من أن لغات البرمجة مثل C و C++ تسمح للمبرمج بتطبيق تعدد العمليات، فإن ذلك الأمر يستغرق الكثير من الجهد مقارنة بـJava.

وهكذا يمكنك تعيين العديد من العمليات للتنفيذ في نفس الوقت. ويمكن أن تؤدي هذه العمليات إلى تحسين التفاعل مع المستخدمين، بحيث يتم التعامل مع العديد من الطلبات في وقت واحد. وكذلك تتم إتاحة الأوقات الخاصة باستجابة الوقت الفعلي.



ومن الأمثلة الجيدة على العمليات التي يتم تنفيذها إمكانية استعادة الذاكرة غير المستخدمة. وهذه الإمكانية هي عملية ذات أولوية منخفضة تنفذ في نفس الوقت مع مهام أخرى. وهي تراقب الكائنات وكل الإشارات إليها، مع التخلص

من الكائنات غير المستخدمة. وهكذا تنفذ العمليات الأخرى في نفس الوقت دون حدوث أي مشكلات أو أعطال.

وهناك مثال آخر على تعدد العمليات وهو البرنامج الذي يدعم الإدخال والإخراج والتفاعل مع المستخدم، مثل أخذ مدخلات صوت الوقت الفعلي من ميكرفون في نفس وقت ضبطها مع صورة فيديو يتم تشغيلها على الشاشة. ففي هذه الحالة، سوف يقوم هذا النظام بمطابقة الصوت مع تحركات شفاه الأشخاص الذين يظهرون على شاشة الكمبيوتر في الوقت الفعلي.

وهكذا سيتخلص هذا البرنامج السمعي المرئي من صعوبة تطابق الصورة مع الصوت، وهي غالبًا أصعب مشكلات البرامج السمعية المرئية.

على الرغم من أنه من الصعب تطوير مثل هذا البرنامج باستخدام Java حاليًا، فإن الإصدارات المستقبلية قد تتسم بسرعة كافية لتقوم بذلك. مع ذلك، يعد هذا مثالاً جيدًا على العمليات التي تتعامل مع أجزاء مختلفة من البرنامج.

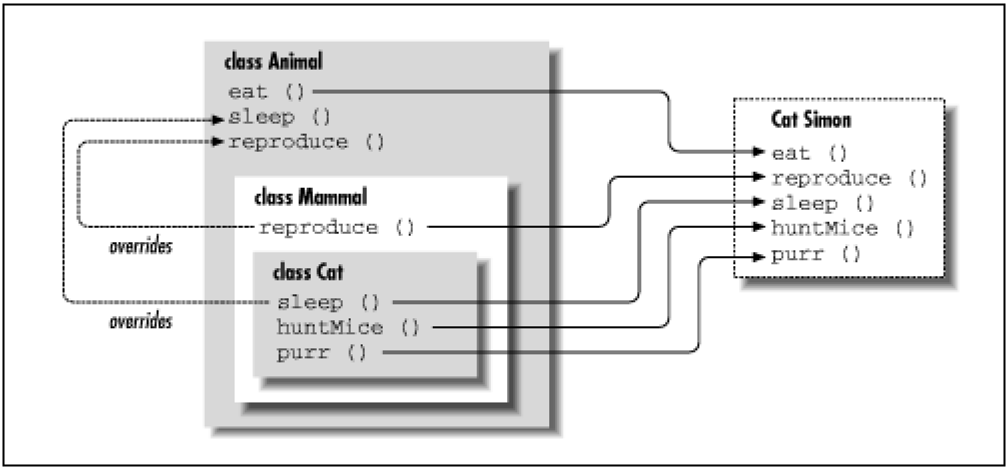
على الرغم من ذلك، وحتى يتم الاستفادة من قوة تعدد العمليات بصورة كاملة، يجب على الأنظمة التي يتم تشغيل كود Java عليها أن تقوم كذلك بدعم العمليات. ونظرًا لأن كل من تعدد المهام الموزع وتعدد العمليات هما الأساس لكل عمليات تطوير نظم التشغيل الحديثة، فإن غالبية نظم التشغيل سوف تكون، في المستقبل القريب، أعلى من المتوسط. أما بالنسبة للوقت الحالي، تم الحد من إمكانيات تعدد العمليات الخاصة بـJava عن طريق هياكل النظم التي يتم عليها تشغيل Java.

لغة ديناميكية:

يجب أن تتسم كل بيئات عمليات المعالجة الموزعة بالديناميكية. وقد تم تصميم Java لتتناسب مع البيئات المتطورة. وهذه اللغة قادرة على دمج الإمكانيات الجديدة، بغض النظر عن المكان الذي أتت منه هذه الإمكانيات - نظام الكمبيوتر المحلي أو الشبكات المحلية أو الشبكات واسعة النطاق أو الإنترنت.

في بيئات C++ التي تدعم استخدام الكائنات، كما في غالبية بيئات البرمجة، لا يكتب المبرمجون كل سطور الكود بأنفسهم. وبدلاً من ذلك، يعتمدون على مجموعة كبيرة من مكتبات الفئات لزيادة إمكانيات البرامج الخاصة بهم. فلو كان المبرمج يريد أن يضيف دعم لإمكانية ما، مثل القدرة على تشغيل صور الفيديو، فإنه ببساطة يدمج مكتبة فئات جديدة يمكنها معالجة هذه المهمة.

في بعض الأحيان، يتم توزيع هذه المكتبات بطرق لا يمكن للمبرمج التحكم بها، مثلما يحدث مع نظام التشغيل أو بيئة النوافذ. في هذه الحالات، ربما تصل عمليات تحديث المكتبات للعميل قبل أن يتاح للمبرمج فرصة إعادة تجميع البرنامج باستخدامها. وعندما يشغل العميل نسخة قديمة من منتج المبرمج، ويحاول أن يدمج مكتبة محدثة مثل هذه، يكون هناك احتمال حدوث المشكلات. ونظرًا لأن البرنامج لم يجمع ثانية للتعامل مع مكتبة الفئات المحدثة، فمن المحتمل أن يتوقف عن العمل.



فلو كانت الفرصة قد أتيحت للمبرمج في بادئ الأمر ليعيد تجميع البرنامج باستخدام مكتبة الفئات المحدثة، ثم يتيح هذا البرنامج للعملاء في الوقت المناسب، لكان من الممكن ألا تحدث مشكلات إطلاقًا. ولذا فإن لغة C++ فشلت في تولي هذه المشكلات.

فإذا قارنا بين إمكانيات C++ المحدودة في هذا المجال وبين لغة Java، نجد أن لغة Java قادرة على دمج مكتبات الفئات الجديدة والعمليات ومتغيرات الكائنات أثناء وقت التشغيل، بدون الخوف من توقف النظام. كما يمكن للغة Java تحديد نوع الفئة وإمكانياتها أثناء وقت التشغيل من خلال الاستعلام، مما يجعل من الممكن دمج فئات جديدة أو إيقاف العملية بأكملها بناء على نتائج هذه الاستعلامات. وهذه الإمكانيات تضفي على برامج Java مستوى من المرونة أثناء التشغيل، الأمر الذي لا يمكن تحقيقه مع غالبية لغات البرمجة الأخرى.

الكلمات الأساسية (المفتاحية) Java Keywords:

إن الكلمات الأساسية لها معنى محدد بالنسبة للمترجم. يوضح الجدول التالي الكلمات الأساسية في لغة Java، لا تعتبر الكلمات true, false, null كلمات أساسية. لا يمكنك استخدام أي من الكلمات السابقة في تسمية الفئات أو الطرق أو المتغيرات. كما يجدر الإشارة أن لغة Java لا تحتوي على معنى محدد لكل الكلمات الأساسية؛ فكلمتي const, goto كلمات أساسية، على الرغم من أنها لم تستخدم حتى الآن.

الكلمات المفتاحية للغة جافا Java keywords				
abstract	assert	boolean	break	Byte
case	catch	char	class	continue
default	do	double	else	extends
final	finally	float	for	If
implements	import	instanceof	int	interface
long	native	new	package	private
protected	public	return	short	static
strictfp	super	switch	synchronized	This
throw	throws	transient	try	Void
volatile	while	const	goto	

الفصل الثاني

مقدمة إلى تطبيقات الجافا

Introduction to Java Applications

برنامجك الأول في لغة Java:

سنبدأ في هذا الفصل بكتابة أول برنامج لنا بلغة Java، وسنعمد إلى كتابة عدة نسخ من هذا البرنامج بهدف التأقلم مع أسلوب كتابة برامج Java، ومن الطبيعي ألا نستطيع من البداية الإحاطة بكل دقائق عناصر اللغة، لذا سنعود لدراسة تلك العناصر بالتفاصيل في الفصول التالية - بإذن الله.

ملحوظة: يجب الانتباه إلى أن لغة Java تميّز بشدة بين الحروف الصغيرة والحروف الكبيرة، لذا ننصح بالتحقق دائمًا من حالة الأحرف في البرامج المكتوبة.

يمكنك البدء بكتابة نص البرنامج التالي باستخدام أي محرر نصوص للغة Java:

```
class Welcome1
{ // main method begins execution of Java application
      public static void main( String [] args )
      {
      System.out.println( "Welcome to Java Programming!" );
      } //  end method main
} // end class Welcome1
```

قم بحفظ هذا البرنامج في ملف يحمل الامتداد java. مثلاً: Welcome1.java. إذ لن يتم ترجمة هذا الملف إذا لم ينته بالامتداد java. كاملة غير منقوصة.

تحليل البرنامج:

يبدأ البرنامج بالكلمة class التي تعني أننا سنعرف فئةً جديدةً لتضاف إلى فئات Java، ويلي هذه الكلمة اسم الفئة والذي هو اسم البرنامج، ويجب أن نأخذ بعين الاعتبار الملاحظات التالية:

- يتضمن كل برنامج فئةً واحدةً على الأقل.
- يجب أن تحمل هذه الفئة اسم الملف ولكن بغير الامتداد java. الموجود في اسم الملف الحاوي.
- تميز لغة Java بين الأحرف الصغيرة والكبيرة، فالكلمتان Welcome وwelcome هما كلمتان غير متطابقتين.
- لا يحمل محرف الانتقال إلى سطر جديد أي معنى في لغة Java، وهو مجرد فاصل بين كلمتين، مثله مثل الفراغ أو علامة الجدولة. ويمكن الفصل بين أي كلمتين باستخدام أي عدد من الفراغات أو محرف الجدولة أو مجرد الانتقال إلى سطر جديد. ويمكن الاستفادة من هذه الميزة في تنسيق البرنامج بحيث يظهر بطريقة واضحة ومرتبة تساعد في فهم الكود، والحالة الاستثنائية الوحيدة هي سطر التعليق حيث يمثل محرف الانتقال لسطر جديد نهاية التعليق.
- يجب أن يأتي بعد اسم الفئة المحرف «}» الذي يحدد بداية كتلة جديدة، ويجب أن تنتهي هذه الكتلة بالمحرف «{».

يحتوي السطر الثاني على تعليق، وهو الذي يبدأ بالمحرف «//» ويمتد التعليق حتى نهاية السطر. والتعليق هو عبارة عن توضيح للمبرمج يتجاهله المترجم. يمكن

استخدام طريقة التعليق متعدد الأسطر وذلك بكتابة المحرفين «*/» في بداية التعليق والمحرفين «/*» عند نهاية التعليق.

السطر الثالث من البرنامج هو السطر الأول في تعريف الفئة، والجزء الأساسي فيه هو:

```
void main ()
```

وهو عبارة عن تعريف لما يسمى طريقة method، والطريقة هي ببساطة وظيفة تنتمي إلى فئة ما، وتأخذ هذه الوظيفة عند تنفيذها وسطاء من أنماط محددة وتعيد قيمة من نمط محدد أيضًا، فالكلمة main تمثل اسم هذه الطريقة أو الوظيفة التي سنعرفها لاحقًا، والكلمة void تعبر عن نمط القيمة التي تعيدها الطريقة، وهي هنا «لا شيء». فالطريقة main هنا لن تعيد عند تنفيذها أية قيمة، وبالتالي فهي لن تستخدم بهدف الحصول على قيمة ما وإنما لما ستؤديه من آثار جانبية ذات صلة بمحيط العمل.

توضع وسطاء الطريقة عادة بين القوسين () ولكل وسيط اسم يسبقه نمط هذا الوسيط، وفي برنامجنا السابق تأخذ الطريقة main وسيطًا أسمه args من النمط String [].

تلميح: -يجري غالبًا، كما سنرى، التصريح عن وسيط الطريقة main بالصيغة:

```
String args []
```

أو

```
String [] args
```

وكلاهما مقبولتين، إلا أنه من الناحية العملية يبدو استخدام الصيغة الأولى أكثر شيوعًا، علمًا بأن الطريقة الثانية تبدو أكثر منطقية.

-وبالمقابل، ليس ضروريًا أن نطلق الاسم args على هذا الوسيط ويمكن استخدام أي اسم شريطة أن لا يبدأ برقم، كما أن طول الاسم غير محدود.

تمثل كل من الكلمتين public وstatic صفةً من الصفات المميزة للطريق، فالصفة public تعني أنه بإمكان مفسّر Java كما بإمكان كل الفئات الأخرى استدعاء هذه الطريقة، وحذف هذه الصفة لا يعني وجود أخطاء قواعدية في البرنامج إلا أنه يحول دون تنفيذ المفسّر لهذه الطريقة، وفي الواقع يمكن القول أن الوظيفة الأساسية للمفسّر هي تنفيذ الطريقة main. ففي حالة غياب الطريقة main يحول ذلك دون تنفيذ البرنامج. ولكن غيابها لا يمثل خطًا وإنما الخطأ في محاولة تنفيذ برنامج كهذا، إذ قد نحتاج لترجمة فئة لا تشكل بحد ذاتها برنامجًا تنفيذيًا بل عنصرًا تستخدمه تطبيقات وبرمجيات أخرى.

أما الصفة static فلها معنى أكثر تعقيدًا سنترك توضيحه إلى فصول لاحقة. ويكفي الآن أن نذكر أن هذه الصفة لا تطلق على كل الطرق.

وبشكل عام يمكن تلخيص ما سبق بحيث يجب أن يتضمن أي تطبيق Java فئةً واحدةً على الأقل:

- تحمل اسم الملف الذي نخزن فيه البرنامج
- تتضمن (على الأقل) طريقة:
  - تتميز بالصفتين public وstatic

- تسمى main
- لها وسيط من النمط [] String

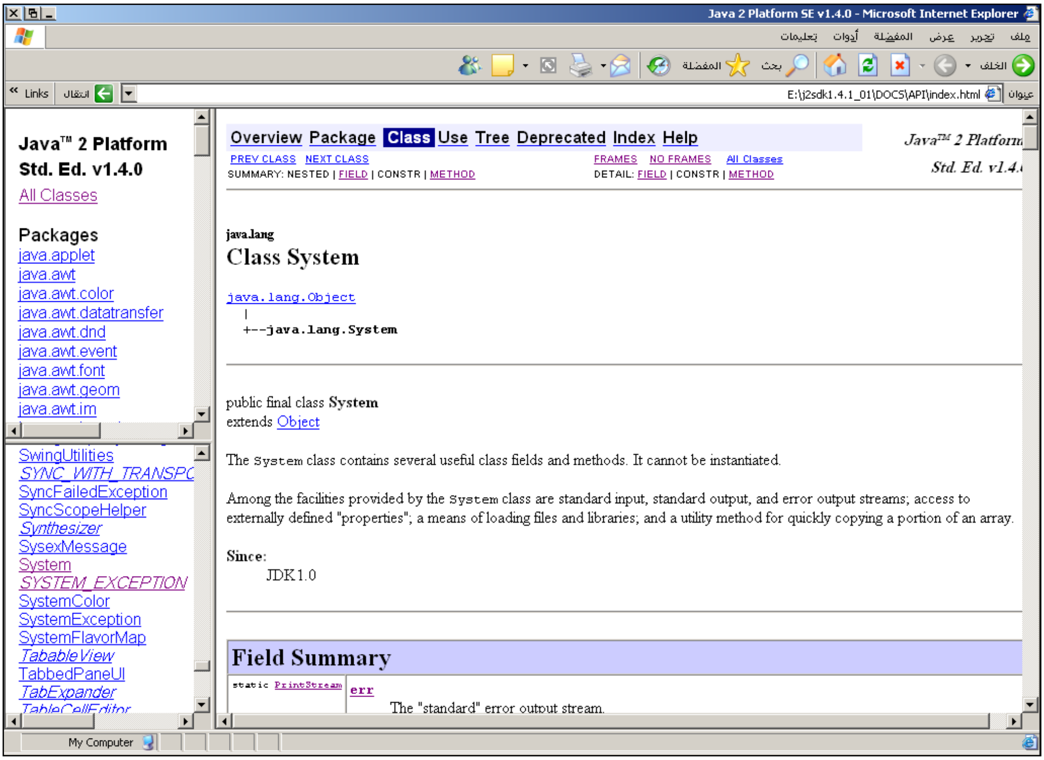
تمتلك كل طريقة جسمًا يبد بالمحرف «}» الذي يحدد بداية كتلة الطريقة، ويجب أن تنتهي هذه الكتلة بالمحرف «{».

السطر الخامس يشكل جسم الطريقة main، وهو يتألف هنا من استدعاء لطريقة مع وسيطها، وهي الطريقة println التي تأخذ وسيطًا من النمط سلسلة محرفية لتظهر قيمته على مجرى الخرج القياسي مسبوقًا بمحرف الانتقال إلى سطر جديد، ويكتب وسيط هذه الطريقة بين القوسين ().

ملحوظة: يجب وضع السلاسل المحرفية أو العبارات النصية بين علامتي اقتباس مزدوجتين "" للدلالة على أن هذه القيمة هي سلسلة محرفية ثابتة.

ولكي يستطيع مفسر Java استخدام الطريقة println يجب أن نرشده إليها، إذ لا يمكن أن تتواجد طرق Java إلا ضمن كائنات Objects، التي يمكن أن تكون فئةً أو أمثالاً instances من فئات، والطريقة println تنتمي للفئة out التي تنتمي بدورها إلى الفئة System.

يمكن الحصول على معلومات كاملة عن فئات Java القياسية بالرجوع إلى جزء التوثيق الخاص باللغة، وذلك بفتح الملف index.html الموجود في المجلد docs/api، ويبين الشكل التالي عرضًا لتوثيق (معلومات) الفئة System:



يجب ملاحظة أن السطر الخامس انتهى بالمحرف «;» فاصلة منقوطة، والتي تشير إلى انتهاء التعليمة، وتعتبر الفاصلة المنقوطة ضرورية جدًا بعد كل تعليمة في برامج Java. ويؤدي غيابها إلى إظهار خطأ قواعدي أثناء الترجمة.

ترجمة البرنامج:

قبل تنفيذ البرنامج السابق، لابد من ترجمته أولاً، ويتم ذلك بتشغيل المترجم java بكتابة الأمر:

```
javac Welcome1.java
```

في حالة وجود أخطاء في البرنامج فإن المترجم يقوم بتوضيح ذلك، بإعطاء رقم السطر الذي يحتوي على الخطأ مع توقعه لسبب الخطأ. وحالما تُصحح أخطاء البرنامج يمكنك القيام بتنفيذه بكتابة الأمر:

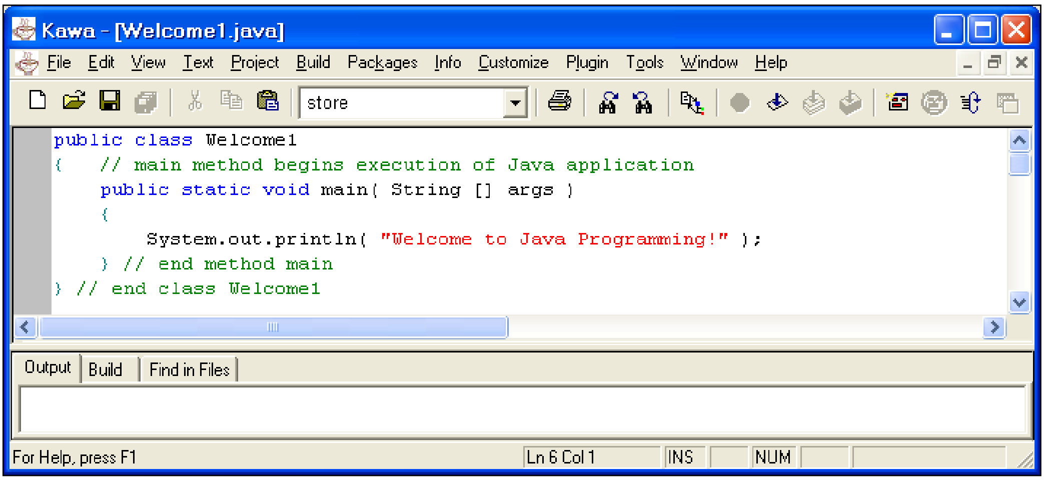
```
java Welcome1
```

ليظهر عندئذٍ السطر المطلوب كالتالي:

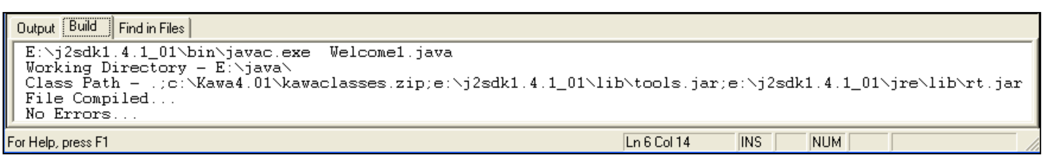


استخدام المحرر Kawa:

عنـد اسـتخدام محـرر نصـوص خـاص باللغـة Java لكتابـة الـبرامج، سـتظهر بعـض الكلمـات بـألوان مميـزة، كـاللون الأزرق للكلمـات المفتاحيـة للغـة Java، والأخضـر للتعليقـات وغيرهـا، ويسـاعد محـرر اللغـة علـى إكتشـاف الأخطـاء الإملائيـة فـور حدوثها.



لترجمـة البرنـامج بعـد حفظـه بالامتـداد.java قـم بـالنقر علـى الأمـر «compile» مـن القائمـة «Build» أو الضـغط علـى المفتـاح «F7» فيـتم الترجمـة لتظهـر الرسـالة التاليـة في حالة خلو البرنامج من الأخطاء:



أمـا في حالـة وجـود أخطـاء فتظهـر رسـالة كالتاليـة والـتي تعـبر عـن حـذف الفاصـلة المنقوطة من السطر الخامس:

```
Output | Build | Find in Files
E:\j2sdk1.4.1_01\bin\javac.exe  Welcome1.java
Working Directory - E:\java\
Class Path - .;c:\Kawa4.01\kawaclasses.zip;e:\j2sdk1.4.1_01\lib\tools.jar;e:\j2sdk1.4.1_01\jre\lib\rt.jar
File Compiled...

--------------------------- Compiler Output ---------------------------
Welcome1.java:5: ';' expected
            System.out.println( "Welcome to Java Programming!" )
                                                               ^
1 error


------------------------------------------------------------------------------
Double click on the line with file name
and error number to locate the error.
Errors occurred during compiling    Ln 5 Col 61    INS    NUM
```

قم بتصحيح البرنامج واعد ترجمته إلى أن يخلو من الأخطاء ثم قم بتشغيله بالنقر على الأمر «Run» من القائمة «Build» أو بالضغط على المفتاح «F4» فيظهر التنفيذ كالتالي:

```
Output | Build | Find in Files
E:\j2sdk1.4.1_01\bin\java.exe   Welcome1
Working Directory - E:\java\
Class Path - .;c:\Kawa4.01\kawaclasses.zip;e:\j2sdk1.4.1_01\lib\tools.jar;e:\j2sdk1.4.1_01\jre\lib\rt.jar
Welcome to Java Programming!
Process Exit...
For Help, press F1    Ln 5 Col 16    INS    NUM
```

أمثلة وحالات للتعامل مع الخرج:

يمكن طباعة العبارة السابقة بعدة طرق، فعلى سبيل المثال البرنامج التالي الذي يستخدم التعليمتين print, println:

```
// Printing one line of text with multiple statements.
public class Welcome2 {
      public static void main( String args [] )
      {
      System.out.print( "Welcome to " );
      System.out.println( "Java Programming!" );
      }
}
```

```
Welcome to Java Programming!
```

كما يمكننا طباعة نص مؤلف من عدة أسطر باستخدام نفس التعليمات السابقة كالتالي:

```
// Printing one line of text with multiple statements.
public class Welcome3 {
      public static void main( String args [] )
      {
      System.out.println(
             "Welcome\nto\nJava\nProgramming\t!" );
      }
}
```

```
Welcome
to
Java
Programming      !
```

يعبر الحرف «\n» عن «السطر الجديد» وتسمى هذه النوعية من المحارف بـ«سلاسل الهروب»، ففي كل مرة تظهر فيها السلسلة «\n» في نص السلسلة المطلوب طباعتها، يتم وضع المؤشر في بداية السطر التالي. فللحصول على سطر فارغ، نقوم بكل بساطة باستخدام حرفي «سطر جديد» خلف بعضهما البعض. سنورد هنا بعضاً من سلاسل الهروب الشائعة في اللغة:

وظيفتها	سلسلة الهروب
سطر جديد، وضع المؤشر في بداية السطر التالي	\n
مسافة الجدولة الأفقية، تحريك المؤشر مسافة ٥ مسافات فارغة للأمام	\t
الانتقال إلى صفحة جديدة	\f
إعادة المؤشر return carriage، وضع المؤشر على بداية السطر الحالي دون التقدم للسطر التالي	\r

سلسلة الهروب	وظيفتها
\b	الحذف الخلفي backspace
\\	الشق المعكوس backslash، تستخدم لطباعة المحرف «\»
\"	علامة اقتباس مزدوجة، تستخدم لطباعة المحرف «"»

عرض النصوص في صناديق الحوار Dialog Boxes:

كانت البرامج السابق تتعامل مع طرق الخرج الأساسية الخاصة بالنظام وهي ما تسمى بـ«واجهة المستخدم النصية CUI Interface User Character» وهو نظام يستخدم شاشة سوداء نصية لإدخال الأوامر إلى الكمبيوتر وهو نظام غير مريح لأنه لا يوفر أي جاذبية للتطبيق.

تمتلك لغة Java نظامًا آخرًا وهو «واجهة المستخدم الرسومية Graphical User Interface GUI» يستخدم مجموعة من الأدوات تتيح للمستخدم التفاعل مع البرنامج مثل النوافذ والقوائم والأزرار. ولبناء واجهة المستخدم الرسومية GUI نحتاج مجموعة مختلفة من الأدوات، وسنتطرق هنا لإنشاء واجهات GUI باستخدام مكتبة أدوات الـSwing، إذ سنستخدم أداة صناديق الرسائل JOptionPane، التي تقوم بإظهار مربع حوار تظهر فيه رسالة تحتوي نصًا نحدده مسبقًا. والمثال التالي لتوضيح ذلك:

```
// Java packages
Import javax.swing.JOptionPane; //program uses JOptionPane

public class Welcome4 {
      public static void main( String args [] )
      {
      JOptionPane.showMessageDialog( null,
                 "Welcome to Java Programming!" );

      JOptionPane.showMessageDialog( null,
```

```
                    "Welcome\nto\nJava\nProgramming!" );

        System.exit( 0 ); //terminate application with window
        }
}
```





التوضيح: تستخدم تعليمة import لاستيراد مكتبة فئات معينة ، ففي برنامجنا قمنا باستيراد الفئة JOptionPane من الحزمة Package المسماة javax.swing وبذلك يمكننا الوصول إلى كل طرق الفئة javax.swing.JOptionPane دون الحاجة لتحديد مسار الوصول كاملاً. فالبرنامج التالي يحقق نفس السابق ولكن دون الحاجة لاستخدام التعليمة import:

```
public class Welcome5 {
      public static void main( String args [] )
      {
      javax.swing.JOptionPane.showMessageDialog( null,
                         "Welcome to Java Programming!" );

      javax.swing.JOptionPane.showMessageDialog( null،
```

```
                        "Welcome\nto\nJava\nProgramming!" );

      System.exit( 0 );
      }
}
```

ملحوظة: -يمكننا استيراد كافة فئات الحزمة كالتالي:

```
import javax.swing.*;
```

-إن كلمة «استيراد» تستخدم هنا بمعنى مجازي، إذ لا تستورد هذه التعليمة شيئًا في الواقع ويمكنك استخدام أي عدد من تعليمات import دون أن يؤثر ذلك على مستوى أداء البرنامج أو على حجمه.

-لا يمكن استيراد أكثر من حزمة في التعليمة ذاتها، فالتعليمة:

```
import java.awt.*;
```

تسهل الوصول إلى كل فئات الحزمة java.awt وليس إلى كل الحزم التي تبدأ أسماؤها بـjava.awt، حيث أنه توجد ١١ حزمة تبدأ بهذا الاسم.

الطريقة showMessageDialog هي إحدى طرق الفئة JOptionPane وتستخدم لعرض رسالة نصية على الشاشة ضمن مربع حوار بسيط، هذه الطريقة تحتاج إلى وسيطين الأول يستخدم لربط مربع الحوار هذا بالنافذة المستدعية وهنا سوف نضعها دائمًا null أي أن هذا المربع غير مرتبط بأي نافذة، والوسيط الثاني عبارة عن النص المطلوب طباعته.

تلميح: الصيغة العامة لتنفيذ أو استدعاء طريقة من فئة معينة تكون كالتالي:

ClssName. methodName (arguments)

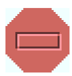
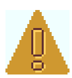
العبارة ;System.exit(0) لإنهاء التطبيق وإغلاق النوافذ.

تخصيص مربع الحوار MessageDialog:

إن الصيغة الأساسية لهذه الطريقة هي كالتالي:

```
void showMessageDialog(Component parentComponent, Object
message, String title, int messageType);
```

الوسيط الثالث هو عنوان مربع الحوار والوسيط الرابع عبارة عن رقم يعبر عن نوع مربع الحوار والجدول التالي يوضح بعضًا من الحالات:

نوع مربع الحوار	الأيقونة	الوصف
JOptionPane.ERROR_MESSAGE		يستخدم لعرض رسالة حرجة أي رسالة تفيد بأنه قد وقع خطأ ما.
JOptionPane.INFORMATION_MESSAGE		تستخدم لعرض معلومات حول شيء ما وهي الحالة الافتراضية.
JOptionPane.WARNING_MESSAGE		تستخدم لعرض رسالة تحذيرية أو تنبيهية مثل رسائل تأكيد تنفيذ عملية معينة.
JOptionPane.QUESTION_MESSAGE		تستخدم عند طلب تأكيد أو قيمة معينة من المستخدم.
JOptionPane.PLAIN_MESSAGE	بدون أيقونة	تستخدم عند عدم الحاجة لإظهار أيقونة في مربع الرسالة.

وسنورد هنا مثالاً يقوم بعمل بعض التخصيص لمربع الرسالة:

```
import javax.swing.JOptionPane;
public class Welcome6 {
      public static void main( String args [] )
      {
            JOptionPane.showMessageDialog(null, "1",
                  " titel", JOptionPane.ERROR_MESSAGE );

            JOptionPane.showMessageDialog(null, "2",
                  "titel",JOptionPane.INFORMATION_MESSAGE );

            JOptionPane.showMessageDialog(null, "3",
                  " titel",JOptionPane.WARNING_MESSAGE );

            JOptionPane.showMessageDialog(null, "4",
                  " titel",JOptionPane.QUESTION_MESSAGE );

            JOptionPane.showMessageDialog(null, "5",
                  " titel",JOptionPane.PLAIN_MESSAGE );
            System.exit( 0 );
      }
}
```











حجز المواقع في الذاكرة وتخصيص قيم لها:

يستخدم البرنامج التالي فكرة حجز مواقع في الذاكرة لتخزين قيم معينة عن طريق ما يسمى بالـ«المتغيرات (المتحولات) Variables»، كما يحتوي البرنامج على طرق إدخال بيانات من المستخدم لتخزينها:

```
import javax.swing.JOptionPane;
public class Addition {
      public static void main( String args [] )
      {
      int number1; // first number to add
      int number2; // second number to add

      String firstNumber; // first string entered by user
      String secondNumber; // second string entered by user

      int sum; // sum of number1 and number2

      // read in first number from user as a string
      firstNumber = JOptionPane.showInputDialog(
                                  "Enter first integer" );

      // read in second number from user as a string
      secondNumber = JOptionPane.showInputDialog(
                                  "Enter second integer" );

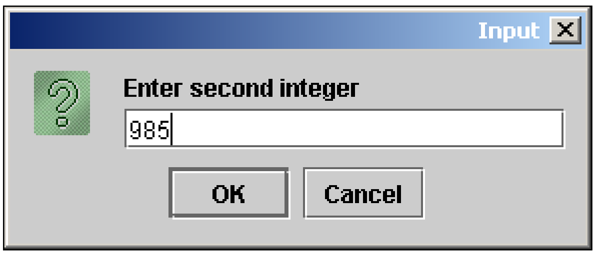
      // convert numbers from type String to type int
      number1 = Integer.parseInt( firstNumber );
      number2 = Integer.parseInt( secondNumber );

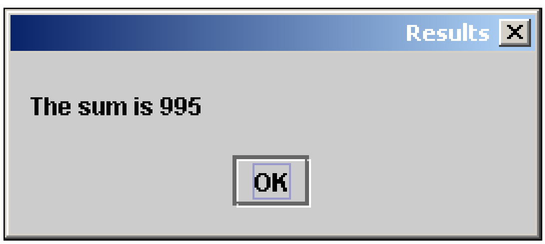
      // add numbers
      sum = number1 + number2;

      // display result
      JOptionPane.showMessageDialog( null, "The sum is " +
            sum, "Results", JOptionPane.PLAIN_MESSAGE );
```

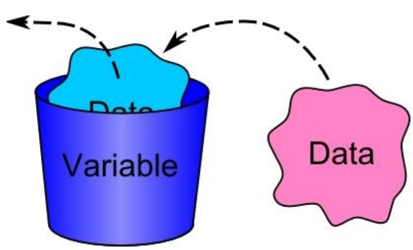
```
        System.exit( 0 );
        }
}
```







يعبر السطر int number1؛ عن طريقة التصريح عن المتغير number1 الذي سيحمل بيانات من النمط int أي عدد صحيح. والمتغير Variable عبارة عن اسم مكان لتخزين البيانات في ذاكرة الكمبيوتر.



نقصد بعبارة «التصريح عن المتغير» عملية تحديد موضع في الذاكرة بحيث يمكن أن نخزن فيه قيمة ليتم استخدامها من قبل البرنامج. يخصص التصريح التالي:

```
int number1;
int number2;
```

المتغيرات number1 وnumber2 على أنها ستخزن بيانات من النمط int وهذا يعني أن تلك المتغيرات سوف تحتوي على قيم صحيحة Integer مثل الأعداد: ٤، ٨٥٤٧. يجب التصريح عن كل المتغيرات بتحديد أسمائها ونمط البيانات التي سوف تخزن فيها وذلك قبل أن يتم استخدامها في البرنامج. باستخدام اسم المتغير في البرنامج، فأنت بذلك تقوم بالإشارة إلى المعلومات المخزنة في هذا الموقع في الذاكرة. يمكن التصريح عن عدة متغيرات من نفس النمط بواسطة عملية تصريح واحدة أو عدة عمليات تصريح، إذ يمكن إعادة كتابة الأسطر الخمسة الأولى من البرنامج كالتالي:

```
int number1, number2, sum;
String firstNumber, secondNumber;
```

النمط String يعبر عن متغيرات سوف تخزن بيانات نصية أي سلاسل محرفية مثل العبارة «welcome».

أسماء المتغيرات:

في لغة Java يجب أن تتم تسمية المتغيرات بناءً على القواعد التالية:

- يمكن لاسم المتغير أن يتكوّن من أي عدد من الأحرف والأرقام وإشارة تحت السطر (_).
- لابد أن يبدأ الاسم بحرف. كما يمكن أن يبدأ بالعلامة تحت السطر (_)، ولكن لا ينصح باستخدامها في أول الاسم.
- حالة الأحرف، كما علمنا سابقًا فإن لغة Java تتأثر بحالة الأحرف، وبالتالي فإن الاسمين count وCount يشيران إلى متغيرين مختلفين.
- لا يمكن استخدام كلمات Java الأساسية كأسماء متغيرات.

كما يمكن أن تتواجد التصريحات في أي مكان من جسم الطريقة بشرط أن يتم التصريح عن المتغير قبل استخدامه. توجد أنماط كثيرة للبيانات سوف نتطرق لها بعد توضيح المثال الحالي.

العبارة التالية:

```
firstNumber = JOptionPane.showInputDialog(
                              "Enter first integer" );
```

تستخدم لعرض مربع حوار يطلب من المستخدم إدخال قيمة ما ليتم تخزينها في المتغير.



يتم تخزين القيمة المدخلة على أنها سلسلة نصية String ولذلك نلاحظ أن نوع المتغير الذي يسبق إشارة الإسناد (=) هو من النمط String. في حالة إسناد قيمة من نمط معين إلى متغير من نمط آخر فإن المترجم يعتبر ذلك خطًا ويقوم بإصدار رسالة الخطأ. بعد النقر على الزر «Ok» يتم تخزين القيمة المدخلة في المتغير المسند إليه. علمًا بأن البيانات هي بيانات نصية أي لا يمكن إجراء العمليات الحسابية عليها. ولذلك فإننا نحتاج إلى تخزين هذه البيانات على متغيرات صحيحة ليتم إجراء العمليات الحسابية عليها، ولكي نخزن هذه القيم في متغير صحيح فإننا نحتاج إلى إجراء عملية تحويل البيانات النصية إلى مقابلها الصحيح، فمثلاً النص «٧» يتم تحويله إلى الرقم ٧. ولإجراء عملية التحويل نستخدم العبارة التالية:

```
number1 = Integer.parseInt( firstNumber );
```

تقوم هذه الطريقة parseInt والموجودة ضمن الفئة Integer بأخذ قيمة نصية وتحويلها إلى المقابل الصحيح لها وتعيد القيمة الصحيحة ليتم تخزينها في المتغير الصحيح number1، بحيث يمكننا إجراء العمليات الحسابية عليها.

تقوم التعليمة:

```
sum = number1 + number2;
```

بحساب مجموع قيمتي المتغيرين number1 وnumber2 وبإسناد نتيجة العملية للمتغير sum باستخدام عملية الإسناد (=).

وأخيرًا يتم طباعة ناتج العملية بالشكل التالي:

```
JOptionPane.showMessageDialog( null, "The sum is " + sum,
               "Results", JOptionPane.PLAIN_MESSAGE );
```

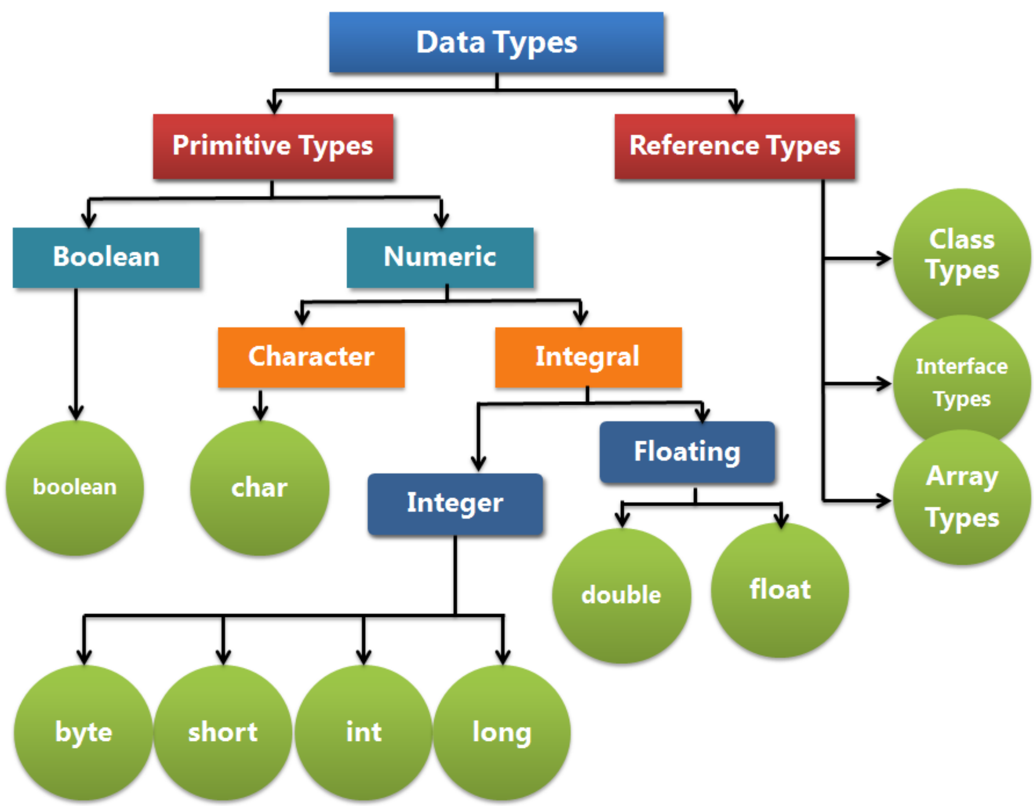
ويجدر الإشارة هنا إلى استخدام الرمز (+) للربط بن السلاسل النصية أو ما يسمى عملية الوصل.

أنماط البيانات في Java:

تمتلك لغة Java مجموعة من أنواع البيانات الأساسية سنوضحها في الجدول التالي:

النوع	القيمة الافتراضية	النطاق المسموح به	الحجم في الذاكرة
byte	0	من -١٢٨ إلى ١٢٧	8 bit's
short	0	من -٣٢٧٦٨ إلى ٣٢٧٦٧	16 bit's
int	0	من -٢١٤٧٤٨٣٦٤٧ إلى ٢١٤٧٤٨٣٦٤٨	32 bit's
long	0l	من -٩,٢٢٣,٣٧٢,٠٣٦,٨٥٤,٧٧٥,٨٠٨ إلى ٩,٢٢٣,٣٧٢,٠٣٦,٨٥٤,٧٧٥,٨٠٧	64 bit's
float	0.0f	(Not-a-Number) NaN وقيم نهائية سالبة ولا نهائي سالب (NEGATIVE_INFINITY) وصفر سالب وصفر موجب وقيم نهائية موجبة ولا نهائي موجب (POSITIVE_INFINITY)	32 bit's

النوع	القيمة الافتراضية	النطاق المسموح به	الحجم في الذاكرة
		من 1.4E-45± إلى 3.40282347E38±	
double	0.0d	NaN ولا نهائي سالب وقيم نهائية سالبة وصفر سالب وصفر موجب وقيم نهائية موجبة ولا نهائي موجب من 4.9E-324± إلى 1.79793134862315 7E308±	64 bit's
char	'\u0000'	من ٠ إلى ٦٥٥٣٥ أو من '\u0000' إلى '\uffff'	16 bit's
boolean	false	false و true	1 bit
void			0 bit



التصريح عن المتغير:

كما أشرنا سابقاً فإن الطريقة القياسية للتصريح عن المتغيرات تكون بالشكل التالي:

```
typename varname1[,varname2[,…]];
```

تعيين القيم للمتغيرات:

الصيغة العامة لتخزين المعلومات في المتغير كالتالي:

```
varname1 = value;
iCounter = 1;
fSalary = 3.121;
bTest = false;
cTime = 'a';
```

الصيغة العامة لتهيئة المتغيرات بقيمة أثناء التصريح:

```
typeName varname1 = value;
int iCounter = 1;
float fSalary = 3.121;
boolean bTest = false;
char cTime = 'a';
```

كما توجد حالة خاصة لتخزين المعلومات في المتغير وهي وقت التشغيل وهذه الحالة تعتمد على وظائف التخزين مثل التي تعرضنا لها فيما سبق والكثير الكثير غيرها:

```
firstNumber = JOptionPane.showInputDialog(
                                "Enter first integer" );
```

العمليات الحسابية Arithmetic:

يوضح الجدول التالي العمليات الحسابية الأساسية في لغة Java:

اسم العملية	الرمز الحسابي	طريقة التعبير الجبري	طريقة التعبير حسب Java
الجمع	+	x+y	x + y
الطرح	-	m-6	m - 6
الضرب	*	2x	2 * x
القسمة	/	x÷3 or $\frac{x}{3}$ or x/3	x / 3
باقي القسمة	%	6 mod 2	6 % 2

يتم تنفيذ العمليات الحسابية حسب أولوياتها حسب الجدول التالي:

العملية	اسم العملية	ترتيب عملية التقييم (الأولوية)
()	الأقواس	تقيم أولاً. إذا وجد في العبارات الحسابية أقواس متداخلة ضمن بعضها البعض فالحساب يبدأ انطلاقاً من أول مجموعة في الداخل. أما إذا كان لدينا مجموعة من الأقواس جانب بعضها البعض وعلى نفس المستوى عندها يبدأ الحساب من اليسار إلى اليمين.
*, / or %	الضرب، القسمة أو باقي القسمة الصحيحة	تقيم ثانياً. وإذا وجدت على نفس المستوى فإنها تقيم من اليسار إلى اليمين.
+ or -	الجمع أو الطرح	تقيم في النهاية، إذا وجدت عدة عمليات منها على نفس المستوى فإنها تقيم من اليسار إلى اليمين.

عمليات المساواة والمقارنة Equality & Relational Operators:

الشكل الجبري	الشكل في Java	مثال على الاستخدام	وصف المثال
=	==	x == y	x تساوي y
≠	!=	x != y	x لا تساوي y
›	>	x > y	x أكبر من y
‹	<	x < y	x أصغر من y
≥	>=	x >= y	x أكبر أو تساوي y
≤	<=	x <= y	x أصغر أو تساوي y

قم بتنفيذ هذا البرنامج الذي يوضح استخدام عمليات المساواة والمقارنة، استعنا في هذا البرنامج بالعبارة if وسوف نتطرق إليها بالتفصيل في الفصول التالية:

```
import javax.swing.JOptionPane;
public class Comparison {
      public static void main( String args [] )
      {
      String firstNumber; // first string entered by user
      String secondNumber; // second string entered by user
      String result=""; // a string containing the output
      int number1; // first number to compare
      int number2; // second number to compare

      //  read first number from user as a string
      firstNumber = JOptionPane.showInputDialog(
                       "Enter first integer:" );

      //read second number from user as a string
      secondNumber  =  JOptionPane.showInputDialog(
                       "Enter second integer:");

      // convert numbers from type String to type int
      number1 = Integer.parseInt( firstNumber );
```

```
      number2 = Integer.parseInt( secondNumber );

      if ( number1 == number2 )
            result = result + number1 + " == " + number2;

      if ( number1 != number2 )
            result = result + number1 + " != " + number2;

      if ( number1 < number2 )
            result = result + "\n" + number1 + " < " +
                  number2;

      if ( number1 > number2 )
            result = result + "\n" + number1 + " > " +
                  number2;

      if ( number1 <= number2 )
            result = result + "\n" + number1 + " <= " +
                  number2;

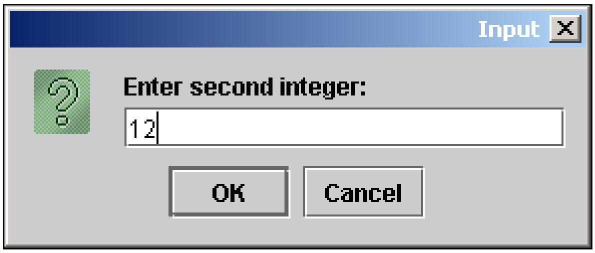
      if ( number1 >= number2 )
            result = result + "\n" + number1 + " >= " +
                  number2;

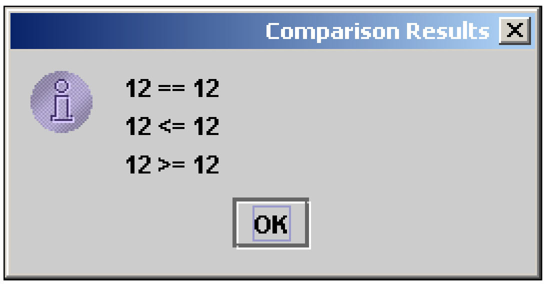
      // Display results
      JOptionPane.showMessageDialog( null, result,
                  "Comparison Results",
                  JOptionPane.INFORMATION_MESSAGE );

      System.exit( 0 );
      }
}
```

يطلب منك البرنامج السابق إدخال قيمتين ثم يجري عليهما عمليات المقارنة، ليطبع في النهاية توضيح لحالات المقارنة الناتجة، فمثلاً التالي:







يجدر الإشارة للعبارة التالية وشبيهاتها:

```
result = result + number1 + " == " + number2;
```

الجزء الذي نريد توضيحه هو الجزء الشبيه بالعبارة:

```
s = s + "abc";
x = x + 5;
```

وهي عبارة عن عملية إسناد قيمة إلى القيمة الحالية في نفس المتغير، فمثلاً إذا اعتبرنا المثال التالي:

```
int x = 1;
x = x + 5;
```

فـإن المـتغير x سـيحمل القيمـة ٦، لأننـا زدنـا علـى قيمتـه المقـدار ٥ ثـم خزنـا النـاتج في المتغير x، الرمز + عند التعامل مع النصوص يعبر عن عملية الوصل.

عمليات الإسناد المختصرة Assignment Operators:

وهـي اختصـار لعمليـات إسـناد قيمـة لقيمـة المـتغير نفسـه والموضـحة في الفقـرة السابقة:

```
s = s + "abc";
x = x + 5;
```

فمـثلاً هـذه الحـالتين يمكـن التعـبير عنهمـا في حالـة عمليـات الإسـناد المختصـرة كالتالي:

```
s += "abc";
x += 5;
```

تقـوم العمليـة =+ بإضـافة قيمـة التعـبير الموجـود علـى يمينهـا إلى قيمـة المـتغير الموجـود علـى يسـارها ثـم تخـزن النتيجـة في المـتغير الموجـود علـى اليسـار. بشـكل عـام يمكـن كتابة أي تعليمة من الشكل التالي إلى تعليمة إسناد مختصر:

```
variable = variable operator expression;
variable operator = expression;
```

ويوضـح الجـدول التـالي عمليـة الإسـناد الحسـابية مـع أمثلـة عليهـا وشـروحها. بفرض أن لدينا:

```
int x = 0, y = 18, z = 3, t = 27, m = 23;
```

عملية الإسناد	مثال	الشكل القياسي	النتيجة
+=	x += 1	x = x + 1	إسناد القيمة ١ إلى المتغير x
-=	y -= 5	y = y - 5	إسناد القيمة ١٣ إلى المتغير y
*=	z *= 2	z = z * 2	إسناد القيمة ٦ إلى المتغير z
/=	t /= 3	t = t / 3	إسناد القيمة ٩ إلى المتغير t
%=	m %= 7	m = m % 7	إسناد القيمة ٢ إلى المتغير m

والبرنامج التالي يوضح استخدام عمليات الإسناد المختصرة:

```
public class Assignment {
      public static void main( String args [] )
      {
            int c = 3;

            c += 5;
            System.out.println( c );

            c -= 2;
            System.out.println( c );

            c *= 4;
            System.out.println( c );

            c /= 8;
            System.out.println( c );

            c %= 3;
            System.out.println( c );
      }
}
```

```
8

6
24
3
0
```

عمليات الزيادة والإنقاص بواحد Increase & Decrement Operators:

يوضح الجدول التالي عمليات الزيادة والإنقاص المتوفرة في لغة Java:

العملية	التسمية	مثال	الوصف
++	عملية الزيادة الأمامية بواحد	++x	زيادة قيمة x بواحد ثم استخدام القيمة الجديدة في العبارة
++	عملية الزيادة الخلفية بواحد	x++	زيادة قيمة x بواحد بعد استخدام القيمة القديمة في العبارة
--	عملية الإنقاص الأمامية بواحد	--x	إنقاص قيمة x بواحد ثم استخدام القيمة الجديدة في العبارة
--	عملية الإنقاص الخلفية بواحد	x--	إنقاص قيمة x بواحد بعد استخدام القيمة القديمة في العبارة

والبرنامج التالي يوضح استخدام عمليات الزيادة والإنقاص:

```
public class Increment
{
      public static void main( String args [] )
      {
            int c = 5;
            System.out.print( c + " " );
            System.out.print( c++ + " " );
            System.out.print( c + " " );
            System.out.println();       // skip a line

            c = 5;
            System.out.print( c + " " );
            System.out.print( ++c + " " );
            System.out.print( c + " " );
            System.out.println();       // skip a line

            c = 5;
```

```
            System.out.print( c + "  " );
            System.out.print( c-- + "  " );
            System.out.print( c + "  " );
            System.out.println();       // skip a line

            c = 5;
            System.out.print( c + "  " );
            System.out.print( --c + "  " );
            System.out.print( c + "  " );
      }
}
```

```
5 5 6
5 6 6
5 5 4
5 4 4
```

تمارين الفصل الثاني:

١ . املأ الفراغات التالية:

- يتم استخدام .... ضمن نص البرنامج من أجل توثيق عمله وتحسين القدرة على قراءته.
- تستخدم الطريقة .... من أجل طباعة البيانات على الشاشة.
- تستخدم التعليمة .... لاستيراد مكتبة فئات إلى برنامجك.
- تستخدم الطريقة .... لقراءة قيمة من المستخدم.

٢ . أكتب تعليمة Java واحدة (أو سطرًا واحدًا) للقيام بالتالي:

- طباعة الرسالة التالية «University of Sciences And Technology»، كل كلمة في سطر.
- اسند قيمة حاصل ضرب المتغيرين x وy إلى المتغير z
- وثق أن هذا البرنامج هو برنامجك الأول
- التصريح عن متغيرين حقيقيين x وy يسند للمتغير الأول القيمة ٣,٥

٣ . بفرض أن x=2 ,y=7. ما الذي سيتم طباعته باستخدام التعليمات التالية:

```
• System.out.println( x + 1 );
• System.out.println( "x + 1" );
• System.out.println( x++ );
• System.out.println( x++ + --y );
• System.out.println( x-- + 1 );
• System.out.println( x += 5 );
• System.out.println( ( x -= 3 ) + y );
• System.out.println( (y *= 2) -= x );
• System.out.println( x -= (y *= 2) );
• System.out.println( "abc" + (y += 5) );
```

```
• System.out.println( "abc" + y += 5 );
```

٤ . قم بتمثيل المعادلات التالية بلغة Java:

- $y = 3x^2 + 2x + 1$
- $y = \frac{2x+2}{x-1}$
- $y \geq 10$
- $y \neq x + x^3$
- $y = x\left(z + \frac{a}{b}\right)^2$

٥ . بين ترتيب تقييم العمليات الحسابية التالية واذكر ما هو الناتج:

```
• x=7+3*6/2-7;
• x=(7+3)*((6/2)-7);
• x=((7+3)*6)/(2-7);
• x=(7+3)*(6/(2-7));
```

٦ . أكتب برنامج يأخذ من المستخدم عددين، ثم يطبع الأكبر بينهما على الشاشة. أما إذا كانا متساويين فيطبع العبارة «there numbers are equal».

٧ . أكتب برنامج يأخذ من المستخدم ثلاثة أعداد صحيحة، ثم يقوم بطباعة مجموعها ومتوسطها وناتج جداءها وقيمة أصغر عدد وقيمة أكبر عدد بينها.

٨ . أكتب برنامجاً يقرأ من المستخدم نصف قطر الدائرة ثم يطبع قيمة قطر الدائرة، محيطها ومساحتها.

٩ . أكتب برنامجاً يقوم بطباعة العبارة التالية على الشاشة:

```
mm\\n p""
a'\**\%d
```

١٠ . أكتب برنامجًا يقرأ من المستخدم عددًا صحيحًا ثم يطبع على الشاشة هل هو عدد زوجي أم فردي.

١١ . أكتب برنامجًا يقرأ من المستخدم عددين ثم يطبع فيما إذا كان العدد الثاني من مضاعفات الأول أم لا.

١٢ . ما الذي يقوم به البرنامج التالي:

```
import javax.swing.JOptionPane;
public class Product {
      public static void main( String [] args )
      {
            int x, y, z, result;
            String xVal, yVal, zVal;

            xVal = JOptionPane.showInputDialog(
                    "Enter first integer:" );

            yVal = JOptionPane.showInputDialog(
                    "Enter second integer:" );

            zVal = JOptionPane.showInputDialog(
                    "Enter third integer:" );

            x = Integer.parseInt( xVal );
            y = Integer.parseInt( yVal );
            z = Integer.parseInt( zVal );

            result = x * y * z;

            JOptionPane.showMessageDialog( null,
                        "The product is " + result );
      System.exit( 0 );
      }
}
```

الفصل الثالث

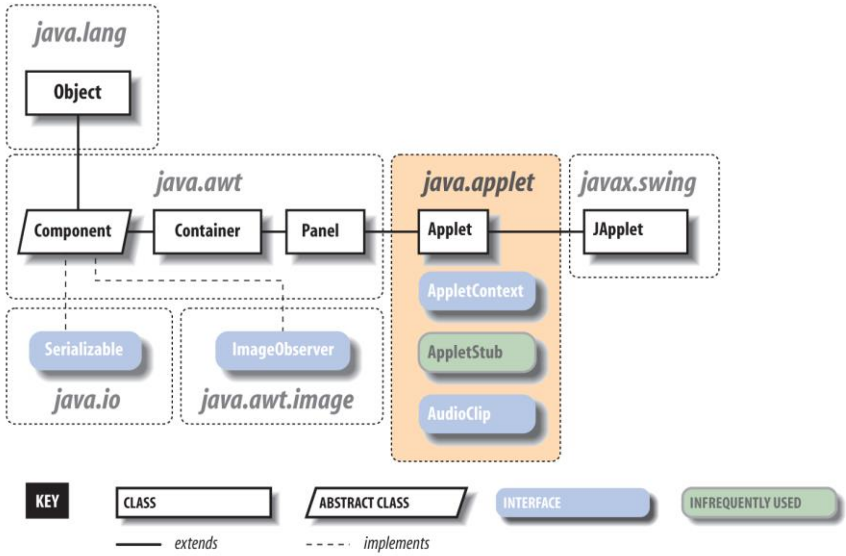
مقدمة إلى بُريمجيات جافا

Introduction to Java Applets

مقدمة:

البُريمج Applet هو برنامج Java مصمم ليقوم بتنفيذه برنامج مستعرض الويب Browser Web. يعتبر كلٌ من تطبيقات Java وبريمجيات Java بمثابة الدعامتين الأساسيتين في برامج Java. كما لاحظنا فيما سبق، تعتبر تطبيقات Java أشبه بتطبيقات سطح المكتب العادية المعروفة، على الرغم من أنها تتطلب وجود بيئة وقت التشغيل الخاصة بلغة Java ليتم استخدامها كما هو الحال في البريمجيات، وعلى النقيض من البريمجيات applets، فإن تطبيقات Java لم يتم تصميمها للعمل على الإنترنت أو ليتم تشغيلها على شبكة الويب. تم تصميم البريمجيات applets للتعامل على الشبكات بحيث يتم تشغيلها كجزء من صفحة الويب. ولهذا، يتم الإشارة إليها على أنها محتويات قابلة للتنفيذ. وعلى الرغم من أنها تعتبر جزءًا لا يتجزأ من صفحات الويب مثل أي محتويات أخرى موجودة فيه، إلا أنها تعتبر بمثابة برامج صغيرة يمكن تشغيلها مثل غيرها.

تتطلب البريمجيات متصفحًا يمكنه التعامل مع لغة Java مثل Netscape Navigator (بداية من الإصدار رقم ٢ وما يليه) أو المتصفح Internet Explorer (بداية من الإصدار رقم ٣ وما يليه) لأنه يجب أن يتم تنزيله من على الإنترنت لإرساله إلى بيئة وقت التشغيل الخاصة بـ Java الخاصة بالمتصفح ليتم تشغيلها.



ملحوظة: تتطلب البريمجيات التي تستفيد من السمات الجديدة الخاصة بـ Java 2 (مثل واجهات الاستخدام الرسومية Swing) برامج ملحقة خاصة بـ Java ليتم تشغيلها لأن بيئة التشغيل الخاصة بمتصفحات الويب لم يتم تحديثها بعد للتعرف على تلك الإمكانيات الجديدة. وتسمح البرامج الملحقة الخاصة بلغة Java والتابعة لشركة Sun لـ applets بأن يتم تشغيلها من قبل Java Run-time Environment (JRE) مع تجاهل البيئة الافتراضية الخاصة بوقت التشغيل في المتصفح.

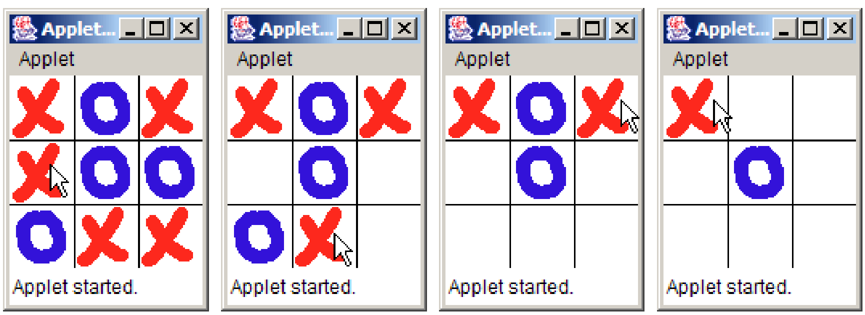
بهذا يمكننا تصنيف الفوارق بين التطبيقات والبريمجيات ضمن فئتين:

- فوارق ناتجة عن طبيعة البريمجيات، من حيث أنها تنفذ انطلاقاً من صفحات الويب.

- فـوارق ناتجـة عـن رغبـات المـبرمجين، وتـدور بشـكل خـاص حـول مبـدأ حمايـة الـبرامج، حيـث أن البريمجيـات تمنـع مـن القـراءة والكتابـة علـى القرص.

أمثلة على البريمجيات وإمكانياتها:

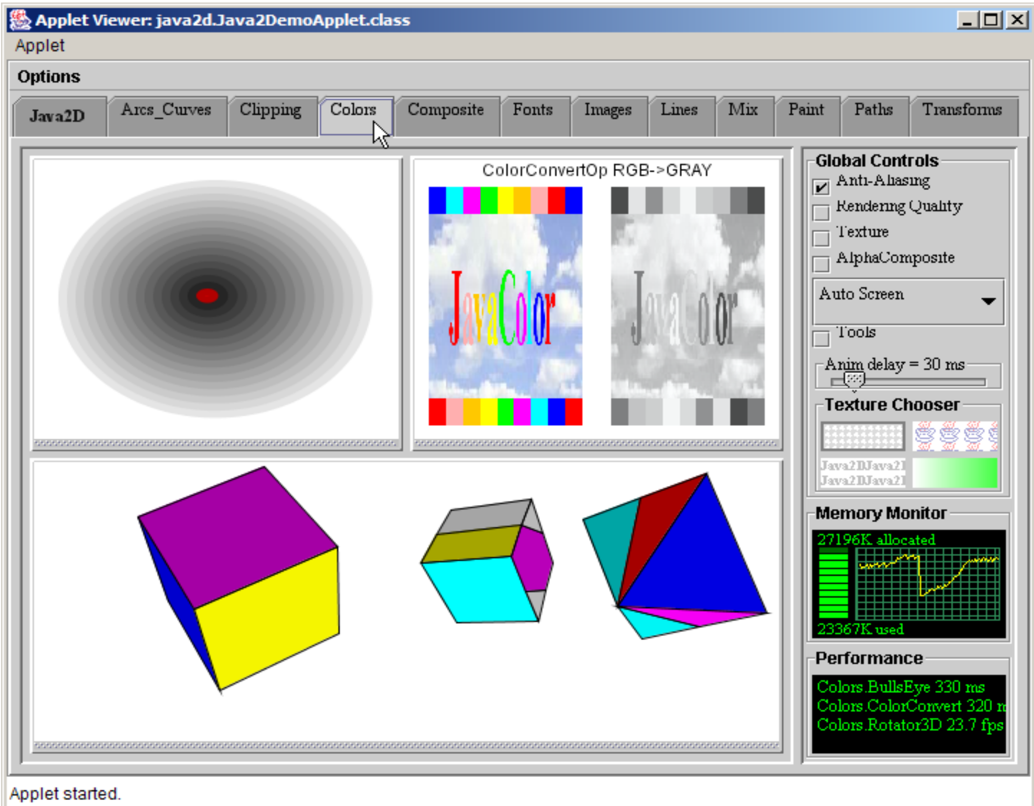
سـنورد هنـا بعضًـا مـن البريمجيـات الـتي تم تصـميمها لأداء مجموعـة مـن الأعمـال المتنوعـة لتتضـح مـدى أهميـة وإمكانيـات البريمجيـات، هنـا نوضـح لعبـة Tic-Tac-Toe تم تصميمها كبريمج Java:



وهذا مثال آخر: عبارة عن بريمج رسم بسيط:



وهذا بريمج للرسوم ثنائية الأبعاد:



البريمج Applet الأول في لغة Java:

سنبدأ بكتابة أول بريمج لنا لتوضيح طريقة عمله وكيفية تشغيله:

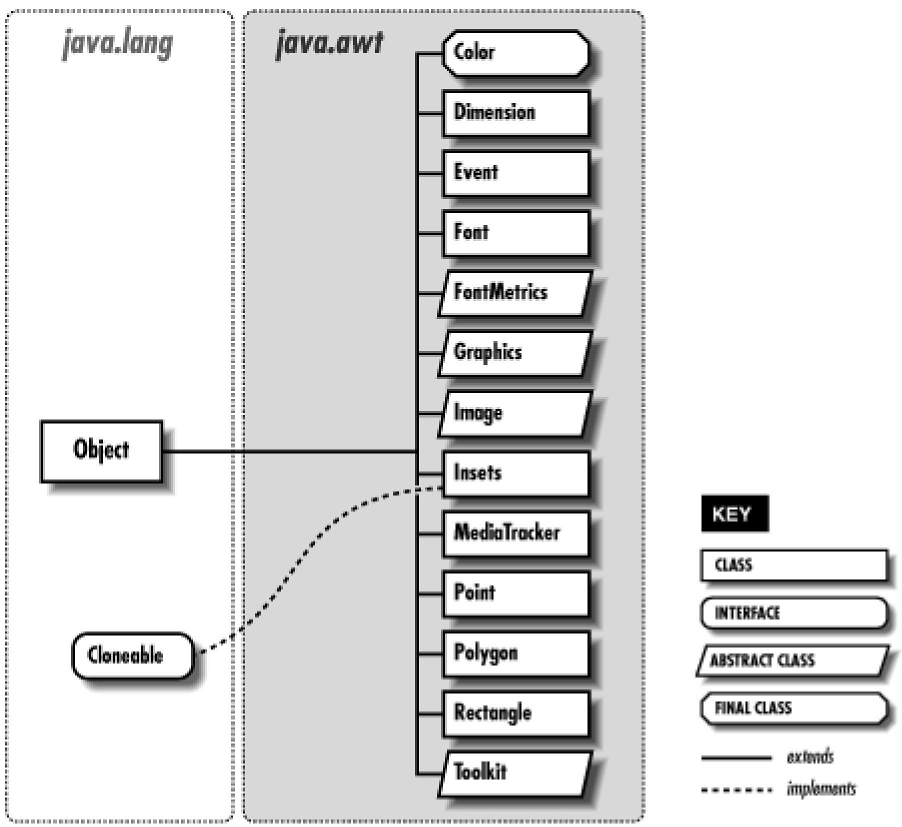
```
// A first applet in Java.
// Java packages
import java.awt.Graphics; // import class Graphics
import javax.swing.JApplet; // import class JApplet

public class WelcomeApplet extends JApplet {
      // draw text on applet's background
      public void paint( Graphics g )
      {
```

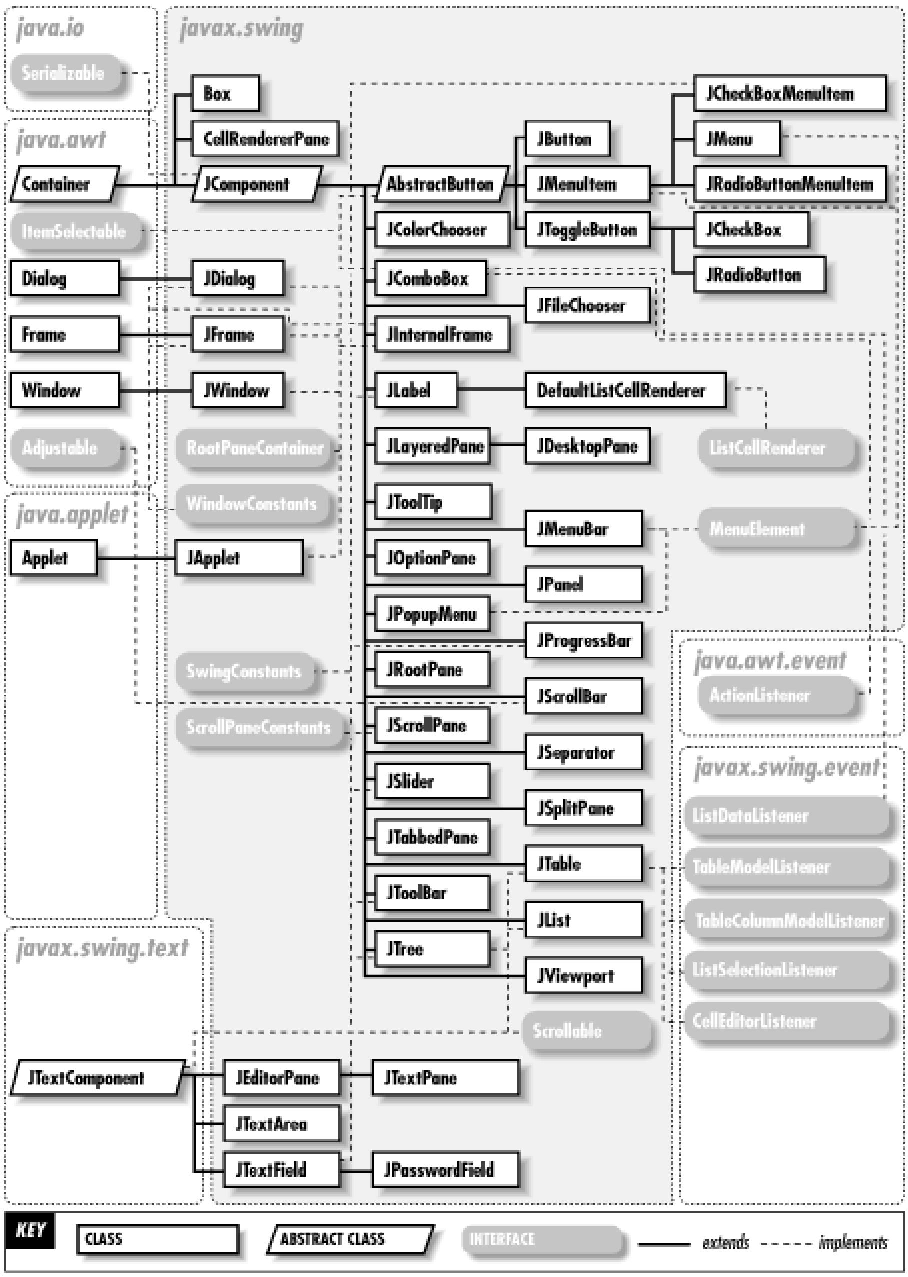
```
      // draw a String at x-coordinate 25
      // and y-coordinate 25
      g.drawString( "Welcome to Java Programming!",
           25, 25 );

      } // end method paint
} // end class WelcomeApplet
```

التوضيح: الهدف الأساسي من هذا البريمج هو طباعة عبارة الترحيب. في البداية قمنا باستيراد الفئة Graphics من الحزمة java.awt حيث تستخدم هذه الفئة في عملية الرسم والتلوين على الشاشة.



واســـتيراد الفئـــة JApplet مـــن الحزمـــة javax.swing وذلـــك لأننـــا ســـنجعل بريمجنـــا كامتداد للفئة JApplet (أي مشتقاً منها).



وذلك كما هو موضح في السطر:

```
public class WelcomeApplet extends JApplet {…}
```

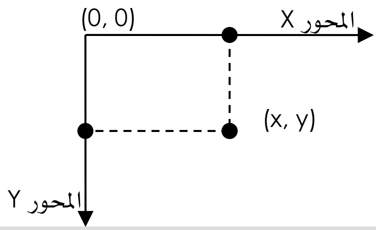
أي أن هذه الفئة ستشتمل على مجموعة من البيانات والعمليات الموجودة في الفئة JApplet والتي تحتوي على مجموعات الفئات التي تسمح بتشغيل كل applet داخل صفحة الويب.

يمتلك البريمج السابق طريقة (تابع) واحدة هي paint والتي تمتلك وسيطًا واحدًا وهو كائن من الفئة Graphics. تعتبر هذه الطريقة هي الأداة الرئيسية في إضافة المحتوى على السطح. إن كائن فئة الرسومات Graphics المُمرر إلى الطريقة paint والمسمى g هو مقبض إلى إطار البريمج. تُعرّف الفئة Graphics أوامر لمعالجة الصور، كالأمر drawstring، يتم إرسال نتائجها إلى الإطار مباشرة. كما يتضمن كائن الرسومات معلومات عن الإطار أيضًا، كحجمه وموضعه ونوع خطه...

لاحظ أن الطريقة paint لا تكتب على الشاشة باستعمال دالة مماثلة لـ println، بل ترسم باستعمال الطريقة drawstring من الفئة Graphics، كما قمنا بتوضيحه في البريمج السابق:

```
g.drawString( "Welcome to Java Programming!", 25, 25 );
```

الوسيط الأول لهذه الطريقة هو سلسلة نصية، بينما الثاني والثالث هما الإحداثيات الأفقي x والعمودي y التي يتم رسم السلسلة عندها. الإزاحة x هي مقياس البعد عن الحافة اليسرى للإطار. والإزاحة y هي مقياس البعد عن حافة أعلى الإطار.



النظام الإحداثي في Java

تنفيذ البريمجيات:

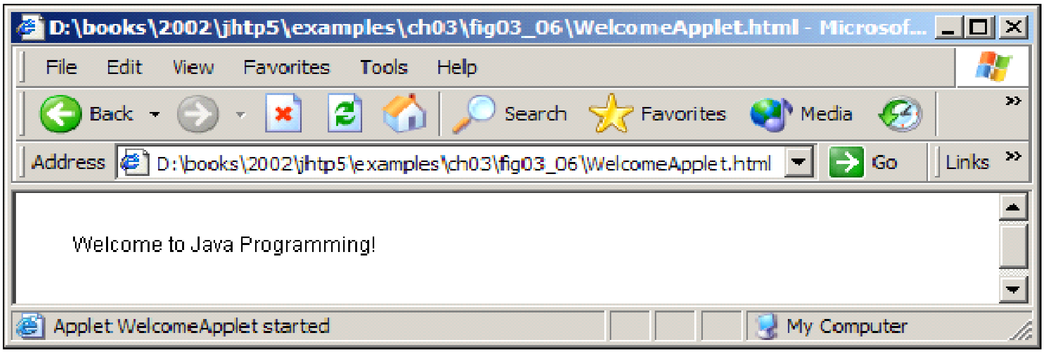
بعد حفظ الكود السابق في ملف قم بترجمته (كما عملنا سابقًا) ليتم إنشاء ملف.class بلغة bytecode. لتنفيذ البريمج من المتصفح، تحتاج إلى إنشاء صفحة HTML (HyperText Markup Language).

أنشئ الملف التالي في أي محرر نصوص:

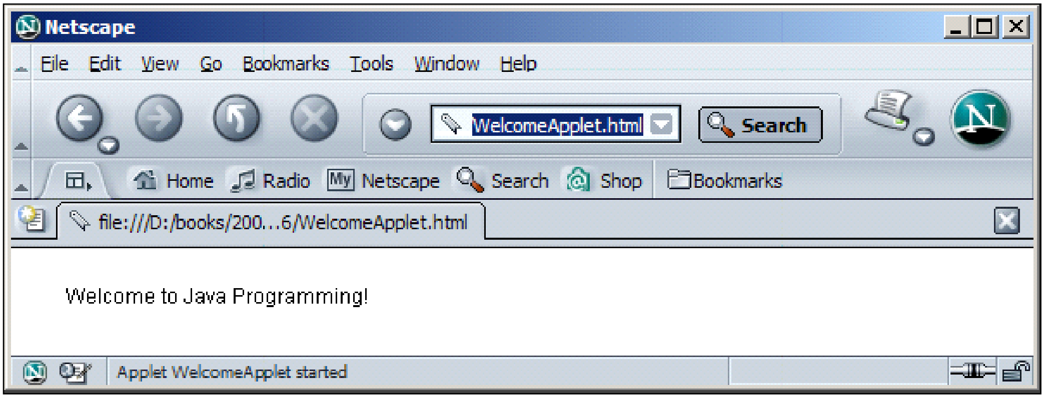
```
<html>
      <applet code = "WelcomeApplet.class" width = "300"
            height = "45">
      </applet>
</html>
```

احفظ هذا الملف بالاسم WelcomeApplet.html (في الواقع، يمكنك هذه المرة استعمال أي اسم تريده طالما كان ينتهي بالامتداد.html؛ فمن غير الضروري أن يتطابق اسم صفحة html مع اسم الفئة).

قم بتشغيل هذا الملف في برنامج مستعرض الويب لديك والداعم للـJava وسيكون المظهر كالتالي:



في المتصفح Internet Explorer

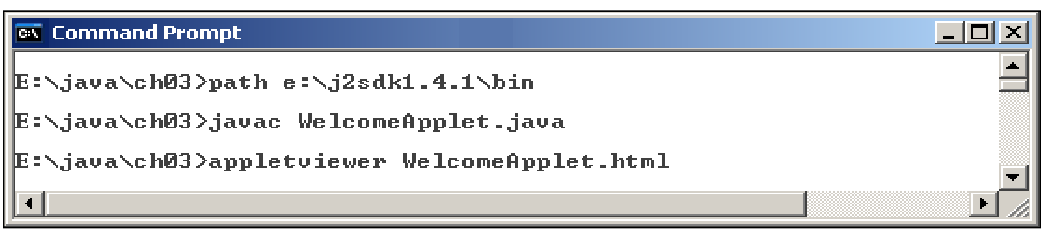


في المتصفح Netscape Navigator

توجد طريقة أخرى لتشغيل البريمج بدون الحاجة لمتصفح الويب وذلك بالاستعانة بالأداة AppletViewer، فبعد كتابة ملف html أكتب السطر التالي:

```
appletviewer WelcomApplet.html
```

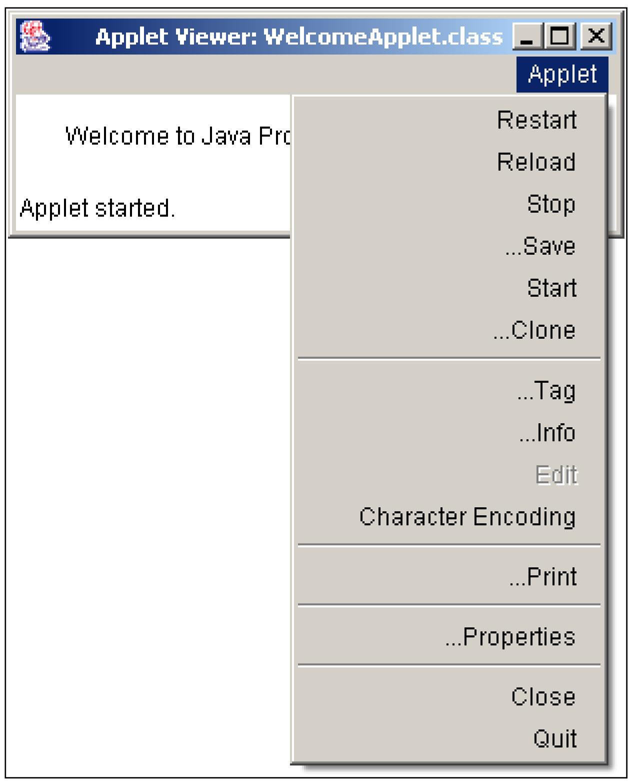
ليتم تشغيل البريمج اتبع الخطوات التالية في موجه النظام:



فتظهر نافذة البريمج التالية:



تمتلك هذه النافذة قائمة تسمى Applet وتستخدم لأداء الأعمال الخاصة بهذا البريمج يمكننا استخدام الأمر Quit لإغلاق النافذة مثلاً.



توضيح: بالنسبة لملف HTML السابق لا يتطلب منك خبرة في لغة HTML ولذا سنوضح بعض الأشياء الهامة التي تعنينا هنا: يشير الوسم (العلامة) <html> إلى بداية شفرة

HTML وهـو يميـز بدايـة صـفحات الويـب، أمـا وسـم نهايـة هـذه الشـفرة فهـو </html>، لإضـافة تعليمـات لتنفيـذ بـريمج Java ضـمن مسـتند html علينـا كتابـة سـطر شـبيه بالتالي:

```
<applet code = "applet.class" width = "300" height =
"45"></applet>
```

حيـث applet.class هـو اسـم الملـف الحـاوي علـى الشـفرة البايتيـة للـبريمج المطلـوب تشـغيله، يجـب الانتبـاه لوسـم النهايـة </applet>. إن الوسـائط الأساسـية الثلاثـة المذكورة في السطر السابق لها وظائف محددة هي:

code: يتم هنا كتابة اسم البريمج المستدعى.

width: عرض المنطقة التي سيتم ضمنها إظهار نتائج البريمج على الشاشة.

height: ارتفاع المنطقة التي سيتم ضمنها إظهار نتائج البريمج على الشاشة.

وهذا مثال آخر على نفس البريمج ولكن باستخدام كائن العنوان Label:

```
import java.awt.Label;
import java.applet.Applet;

public class WelcomeApplet extends Applet {
      public void init( )
      {
            Label strlbl = new Label(
                  "Welcome to Java Programming!" );
            add( strlbl );
      }
}
```



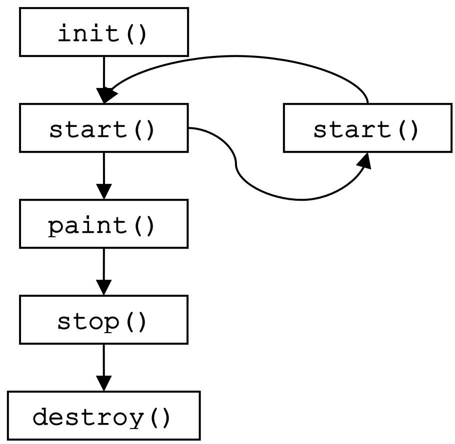
دورة حياة البريمج:

يحتوي البريمج على طرق methods يتم استدعاؤها تلقائيًا من قبل المتصفح، فمثلاً إذا أراد المبرمج إضافة بعض العناصر التي هي من فئة مكونات واجهات المستخدم User Interface Components فلابد عندها من كتابة الطريقة init() أثناء البرمجة حيث ستحتوي على القيم البدائية لهذه العناصر. ويتم تنفيذ هذه الطريقة في بداية عمل البريمج، كما توجد الطريقة start() والتي تنفذ بعد الطريقة init() مباشرة، وفي كل مرة ينتقل فيها البريمج من الحالة غير الفعالة إلى الحالة الفعالة يتم تنفيذ الطريقة start().

توضيح: عند استخدام البرنامج appletviewer لتنفيذ البريمج، عند القيام بتصغير نافذة العرض يدخل البريمج في حالة عدم الفاعلية ولكن عند استعادة النافذة (تكبيرها) سيعود البريمج إلى الحالة الفعالة. أما في حالة الإنترنت فقد يترك المستخدم إحدى الصفحات الفعالة حاليًا ويذهب إلى أخرى وفي هذه الحالة يكون البريمج الموجود في الصفحة السابقة قد دخل في حالة اللافعالية ولكن عند العودة إليها سينشط البريمج مرة أخرى.

ومن الطرق أيضًا الطريقة stop() التي يمكن وضعها ضمن الشفرة في حال لم يرغب المبرمج أن تكون دورة حياة البريمج طويلة. كما يتم تنفيذ هذه الطريقة

تلقائيًا عندما يدخل البريمج في حالة اللافعالية. هناك طريقة رابعة تدعى destroy() وهذه الطريقة تستدعى أيضًا تلقائيًا في حال تم الخروج نهائيًا من الصفحة (إغلاق الصفحة)، أو عند إغلاق نافذة appletviewer. ولا ننسى الطريقة paint() التي تستدعى عند رسم الإطار وعند إعادة التحميل.



دورة حياة البريمج

رسم الخطوط واستخدام سلاسل الهروب:

يوضح المثال التالي استخدام سلاسل الهروب ضمن البريمج:

```
// Displaying multiple strings in an applet.
import java.awt.Graphics; // import class Graphics
import javax.swing.JApplet; // import class JApplet

public class WelcomeApplet2 extends JApplet {
      // draw text on applet's background
      public void paint( Graphics g )
      {
            // draw two Strings at different locations
            g.drawString( "Welcome to", 25, 25 );
            g.drawString( "Java Programming!", 25, 40 );
```

```
    } // end method paint
} // end class WelcomeApplet2
```



يوضح المثال التالي استخدام وظائف رسم الخطوط:

```
// Displaying text and lines.
import java.awt.Graphics; // import class Graphics
import javax.swing.JApplet; // import class JApplet

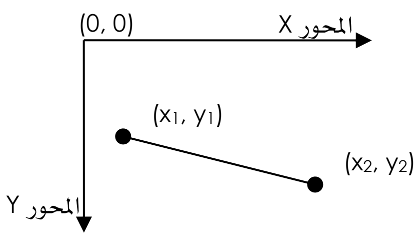
public class WelcomeLines extends JApplet
{
      // draw lines and a string on applet's background
      public void paint( Graphics g )
      {
      // draw horizontal line from (15, 10) to (210, 10)
            g.drawLine( 15, 10, 210, 10 );

      // draw horizontal line from (15, 30) to (210, 30)
            g.drawLine( 15, 30, 210, 30 );

      // draw String between lines at location (25, 25)
            g.drawString( "Welcome to Java Programming!",
                          25, 25 );
      } // end method paint
} // end class WelcomeLines
```



توضيح: تمتلك الطريقة drawLine أربعة وسائط (x_1, y_1, x_2, y_2) يمثل الاثنين الأولين إحداثيات نقطة البداية (x_1, y_1)، أما الوسيطين التاليين فيمثلان إحداثيات نقطة النهاية (x_2, y_2).



نظام الإحداثي الخاص بتمثيل الخط المستقيم

استخدام البريمجيات في التعامل مع أعداد النقطة العائمة:

كنا في الفصل السابق قد تعرضنا لتطبيق يقوم بجمع عددين صحيحين سوف نقوم هنا بعمل بريمج لأداء نفس الوظيفة ولكن على عددين حقيقيين:

```
// Adding two floating-point numbers.
// Java packages
import java.awt.Graphics; // import class Graphics
import javax.swing.*; // import package javax.swing

public class AdditionApplet extends JApplet
{
      double sum; // sum of values entered by user
```

```
    // initialize applet by obtaining values from user
    public void init()
    {
          String firstNumber;
          // first string entered by user

          String secondNumber;
          // second string entered by user

          double number1; // first number to add
          double number2; // second number to add

          // obtain first number from user
          firstNumber = JOptionPane.showInputDialog(
                "Enter first floating-point value" );

          // obtain second number from user
          secondNumber = JOptionPane.showInputDialog(
                "Enter second floating-point value" );

          // convert numbers from type String
          // to type double
          number1 = Double.parseDouble( firstNumber );
          number2 = Double.parseDouble( secondNumber );

          // add numbers
          sum = number1 + number2;
    } // end method init

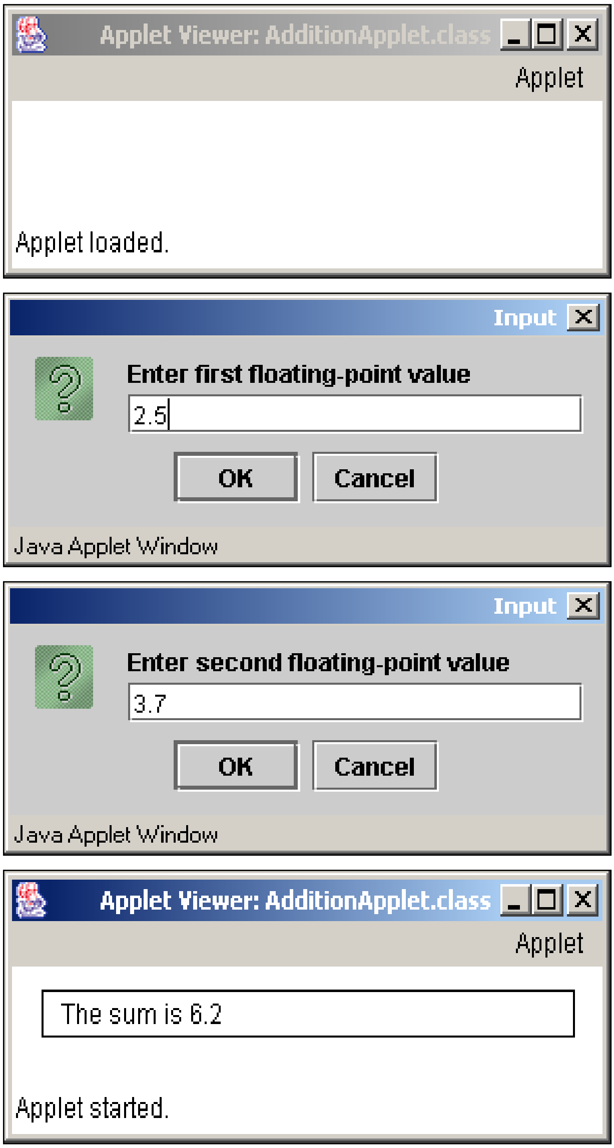
    // draw results in a rectangle on applet's background
    public void paint( Graphics g )
    {
    // draw rectangle starting from (15, 10) that is 270
    // pixels wide and 20 pixels tall
          g.drawRect( 15, 10, 270, 20 );

    // draw results as a String at (25, 25)
```

```
            g.drawString( "The sum is " + sum, 25, 25 );

      } // end method paint
} // end class AdditionApplet
```

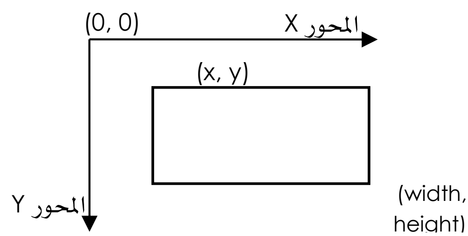
وعند تنفيذ البريمج يظهر التالي:



التوضيح: في هذا البريمج تم استيراد الحزمة javax.swing لأننا سنستخدم الوظيفة showInputDialog الموجودة ضمن الفئة JOptionPane لقراءة بيانات من المستخدم.

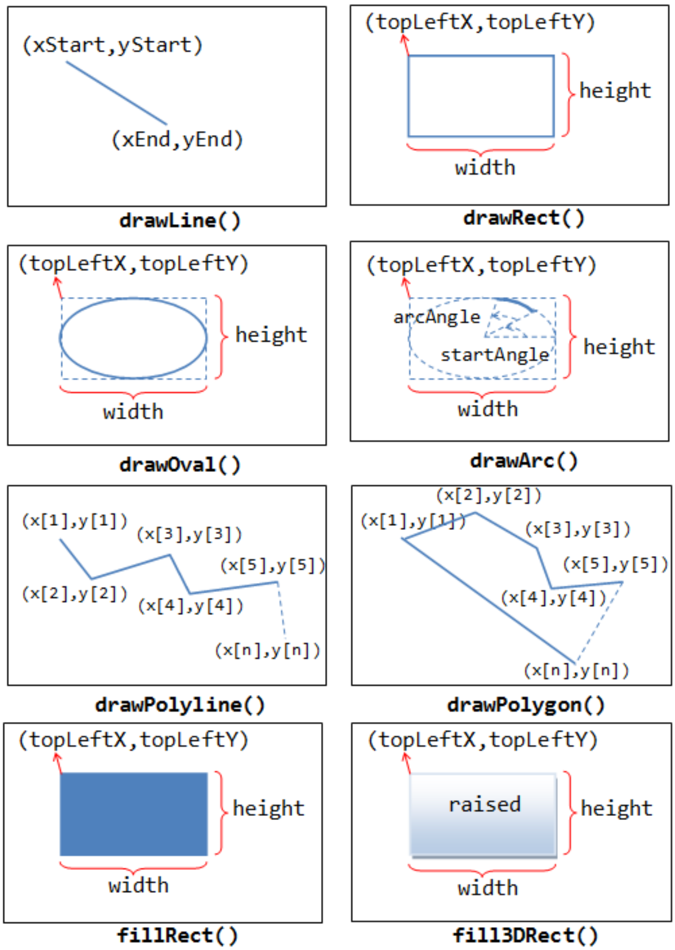
تستخدم الوظيفة Double.parseDouble() لتحويل قيمة نصية إلى قيمة حقيقية (أعداد ذات نقطة عائمة floating-point).

يتم استدعاء الوظيفة init() تلقائيًا عند بدأ تحميل البريمج ويليه يتم استدعاء الطريقة paint() لرسم الإطار مع محتوياته. وقد استخدمنا هنا وظيفة رسم المستطيل drawRect والتي تأخذ أربعة وسائط (x, y, w, h) يمثل الأولين إحداثيات نقطة بداية الرسم، والتاليين يمثلان قيمة عرض وطول المستطيل بالبكسل.



نظام الإحداثي الخاص بتمثيل المستطيل

والشكل التالي يوضح فكرة بعض الطرق الخاصة برسم الأشكال الهندسية:



تنفيذ البريمج بطريقة أبسط:

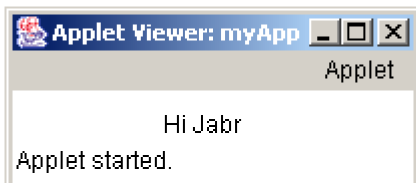
تتطلب الطريقة التي استخدمناها إنشاء ملف HTML من أجل كل بريمج نريد تنفيذه، وهو أمر ممل نوعًا ما فإذا كنت تستخدم برنامج appletviewer يمكنك الاعتماد على إحدى خواصه لتسهيل عملك، فما يهم هذا البرنامج من الملف HTML هو الوسم <applet> ويتجاهل كل شيء ما عداها، ولاستخدام هذه الميزة يمكنك أن تضع في بداية بريمجك سطر تعليقات يتضمن كل وسومات HTML الضرورية كما في المثال التالي:

```
// <applet code = "myApp" width = "200" height = "150">
// </applet>
public class myApp extends java.applet.Applet {
      public void init() {
            add( new java.awt.Label("Hi Jabr") );
      }
}
```

ولتنفيذه قم بترجمته ثم اكتب السطر التالي أمام المحث:

```
appletviewer myApp.java
```

فيظهر الخرج كالتالي:



يجب الانتباه إلى أن هذه الطريقة تستخدم أثناء البرمجة فقط وليس عند إرسال البريمج للمستخدم.

تمارين الفصل الثالث:

١ . ما هو الهدف الأساسي من تصميم البريمجيات Applets؟

٢ . ما هو الفرق بين البريمجيات والتطبيقات؟

٣ . **املأ الفراغات التالية:**

- تزودنا الفئة.................. بمجموعة من طرق الرسم.
- عند بداية تشغيل البريمج فإنه يتم استدعاء الطرق التالية :.................. و.................. و..................
- تستخدم الكلمة.................. لتعريف متغير حقيقي مفرد الدقة، أما الكلمة.................. فتستخدم لتعريف متغير حقيقي مزدوج الدقة.
- لتنفيذ البريمج ضمن مستعرض الويب نحتاج لإنشاء..................
- يستخدم الوسم.................. لتحديد لتعريف بريمج ضمن صفحة الويب (ملفات HTML).

٤ . قم بكتابة بريمج يطلب من المستخدم إدخال أربعة أرقام تمثل إحداثيات نقطتين (x_1, y_1) و(x_2, y_2)، ثم يقوم البريمج برسم خط مستقيم وشكل مستطيل بناءً على هذه الإحداثيات.

٥ . قم بكتابة بريمج يطلب من المستخدم إدخال رقمين حقيقيين، ثم يقوم بحساب مجموعهما وحاصل جدائهما ومتوسطهما، وأيهما أكبر وأيهما أصغر.

٦ . قم بكتابة بريمج يطلب من المستخدم إدخال إحداثيات نقطة (x, y) كمركز لدائرة ثم يدخل قيمة نصف قطر الدائرة. يقوم هذا البريمج برسم هذه الدائرة (استخدم الوظيفة drawOval (x, y, w, h))، ثم يقوم بحساب الآتي:

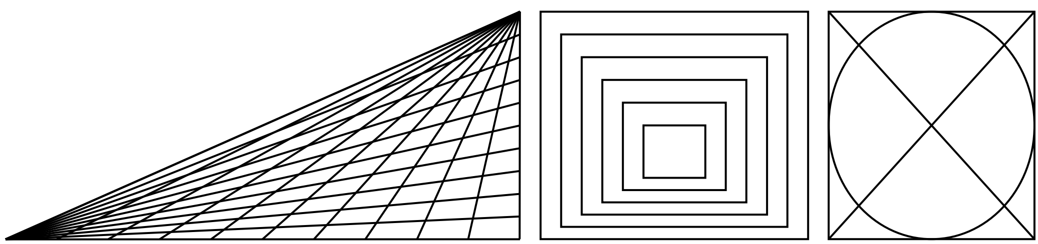
diameter =2r (القطر)

circumference = 2πr (المحيط)

area =πr^2 (المساحة)

ملحوظة: يمثل العدد π القيمة ٣,١٤١٥٩ بالتقريب، كما يمكننا الحصول عليه من الفئة Math بالشكل التالي Math.PI، توجد هذه الفئة في الحزمة java.lang.

٧ . قم بكتابة بريمج لرسم كلٍ من ما يلي على حده:



الفصل الرابع

بُنى التحكم

Control Structures

مقدمة:

قبل كتابة أي برنامج لحل مسألة ما ، يجب أن يتوفر لدينا فهمًا شاملاً للمسألة المطروحة وأسلوبًا منهجيًا للحل. وعند كتابة البرنامج، يجب علينا فهم طبيعة وأنواع الأقسام المؤلفة له كما يجب علينا استخدام مبادئ وطرق مبرهن عليها مسبقًا. سوف نتطرق في هذا الفصل - بإذن الله - لمناقشة جميع القضايا المتعلقة بالمبادئ والطرق الخاصة بالبرمجة المهيكلة. والتي تفيد في عملية فهم وتنفيذ طرق البرمجة كائنية التوجه.

الخوارزميات Algorithms:

يجري حل أي مسألة برمجية من خلال تنفيذ سلسلة من الأفعال وفق ترتيب معين. نطلق تسمية الخوارزمية Algorithm على أية إجرائية لحل مسألة ما تقوم بتحديد:

١ . الأفعال الواجب تنفيذها.

٢ . الترتيب الواجب إتباعه من أجل تنفيذ الأفعال السابقة.

وهذا مثال: لتوضيح خوارزمية الاستيقاظ:

١ . النهوض من السرير

٢ . خلع لباس النوم

٣ . ارتداء ملابس أخرى

٤ . أخذ الحمام الصباحي

٥ . تناول الفطور

٦ . الذهاب للعمل

ويتضح في هذا المثال أهمية تحديد الرتيب الواجب إتباعه من أجل تنفيذ الأفعال المرتبطة بمسألة ما.

لغة الخوارزميات (الشفرة الزائفة) Pseudocode:

لغة الخوارزميات هي عبارة عن لغة مصطنعة وغير متعارف عليها تساعد المبرمجين على تطوير الخوارزميات. تشبه لغة الخوارزميات لغة التخاطب العادية مثل اللغة العربية فهي لغة مبسطة ومفهومة ولكنها ليست لغة برمجة فعلية.

لا يمكن تنفيذ البرامج المكتوبة بلغة الخوارزميات على الحاسب، لكنها تساعد المبرمج على التفكير ببرنامجه قبل محاولة كتابته بأية لغة برمجة.

تتألف لغة الخوارزميات عادةً من مجموعة من الأحرف والكلمات، يمكن تحويل أي نص مكتوب بعناية بلغة الخوارزميات بسهولة إلى النص الموافق له بلغة Java حيث يمكن أن نستبدل في كثير من الحالات التعليمات الخاصة بلغة الخوارزميات بمكافئتها في لغة Java.

بنى التحكم Control Structures:

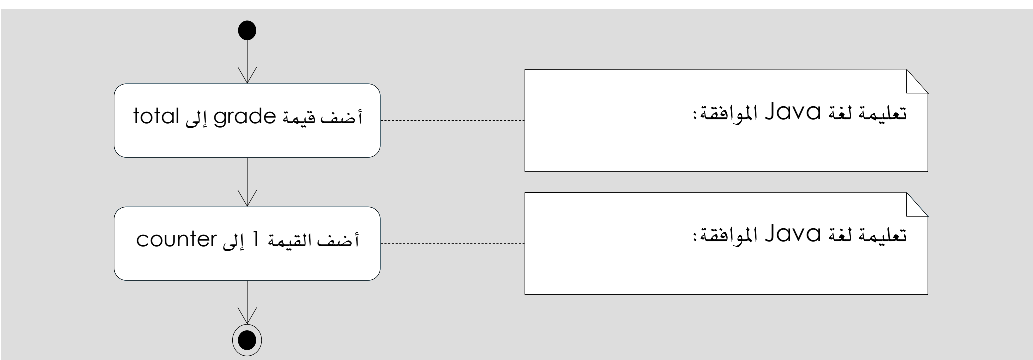
يتم عادةً تنفيذ تعليمات برنامج ما واحدةً تلو الأخرى حسب ترتيب ورودها في نص البرنامج. نسمي أسلوب التنفيذ هذا بـ«التنفيذ التسلسلي Sequential Execution».

برهنت الأبحاث بأنه من الممكن كتابة البرامج باستخدام ثلاثة بنى للتحكم فقط وهي:

- البنية التسلسلية Sequence Structure
- بنية الاختيار Selection Structure
- بنية التكرار Repetition Structure

البنية التسلسلية Sequence Structure:

تتبع كافة لغات البرمجة مثل Java البنية التسلسلية حيث أن الحاسب يقوم بتنفيذ تعليمات اللغة واحدةً تلو الأخرى، حسب ترتيب كتابتها ضمن نص البرنامج. يوضح مخطط UML التالي بنية تسلسلية لتنفيذ عمليتين حسابيتين متعاقبتين. تسمح لنا لغة Java باستخدام أي عدد نريد من الأفعال ضمن البُنى التسلسلية. كما أنه يمكننا أن نستبدل أي فعل بمجموعة من الأفعال ضمن أي بنية تسلسلية كما سنوضح ذلك لاحقاً.



مخطط النشاط لبنية تسلسلية خاصة بلغة Java

لغة النمذجة الموحدة UML Universal Modeling Language:

هي لغة نمذجة مرئية عامة الأهداف تستخدم لتعريف ومعاينة وبناء وتوثيق مكونات النظام البرمجي المنشأة. تهتم نماذج هذه اللغة بتبيان بعض المفاهيم المتعلقة بالنظام وتتجاهل مفاهيم أخرى لتهتم بها النماذج الأخرى، وبحيث تعطي هذه النماذج مجتمعة وصفاً كاملاً للنظام.

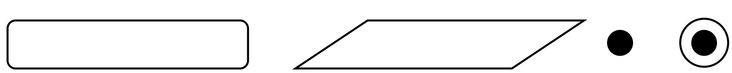
يمكن تصنيف نماذج لغة UML في ثلاث فئات:

- ❖ نماذج الحالة State models (وهي تصف بنى المعطيات)
- ❖ نماذج السلوك Behavior models (وهي تصف علاقات الأفعال بين الكائنات)
- ❖ نماذج تغير الحالة State Change models (وهي تصف حالات النظام الممكنة عبر الزمن)

تتضمن لغة UML عدداً قليلاً من المفردات البنيوية التي تسمح بتجزئة النظام بحيث يمكن تطويره بطريقة تكرارية تزايدية.

تشكل مخططات العمل أو النشاط Activity Diagrams جزءاً من لغة UML. يمثل مخطط العمل تدفق الأعمال Workflow (التي تسمى أيضاً بالنشاط Activity) ضمن جزء من أجزاء النظام البرمجي المدروس.

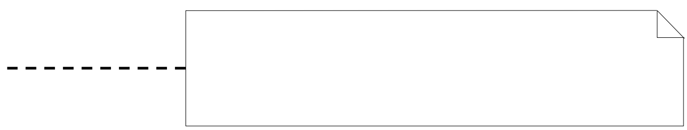
تتكون مخططات النشاط من رموز خاصة ذات معنى مثل رمز حالة الفعل action-state symbols (وهو عبارة عن مستطيل ذو جانبين مقوسين يميني ويساري) ورمز المعين diamond والدوائر الصغيرة smallcircles.



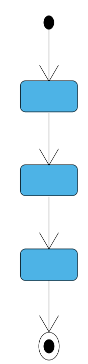
تمثل الدائرة الصغيرة الممتلئة الموجودة عادةً في بداية مخطط النشاط، الحالة الابتدائية للنشاط initial state التي يمر بها قبل البدء بتنفيذ النشاط في حين تمثل الدائرة الصغيرة الممتلئة والمحاطة بحلقة إضافية الموجودة في نهاية مخطط النشاط، الحالة النهائية التي يصل إليها النشاط في نهاية تنفيذه.

يجري الربط بين هذه الرموز بواسطة أسهم الانتقال Transition assows التي تمثل الحركة والتدفق ضمن النشاط.

كما هو الحال مع لغة الخوارزميات، تساعد مخططات النشاط على تطوير وتمثيل الخوارزميات. ومن الواضح أن هذه المخططات تظهر جريان التحكم بين الأجزاء المختلفة المكونة للنظام المدروس.



أما بالنسبة للمستطيلات ذات الزاوية العلوية اليمنى المثنية، فتسمى بالملاحظات note في لغة UML وهي تفيد في إيراد شروح إضافية حول الهدف في وجود الرمز في المخطط. يمكن استخدام الملاحظات في أي مخطط من مخططات UML وليس مخططات النشاط فقط. استخدمنا الملاحظات في المخطط السابق لتوضيح تعليمات لغة Java المقابلة التي تعبر عن النشاط. نلاحظ أيضًا وجود خط متقطع للربط بين الملاحظة والرمز الذي تعبر عنه.



المخطط الصندوقي للبنية التسلسلية

بنية الاختيار Selection Structure:

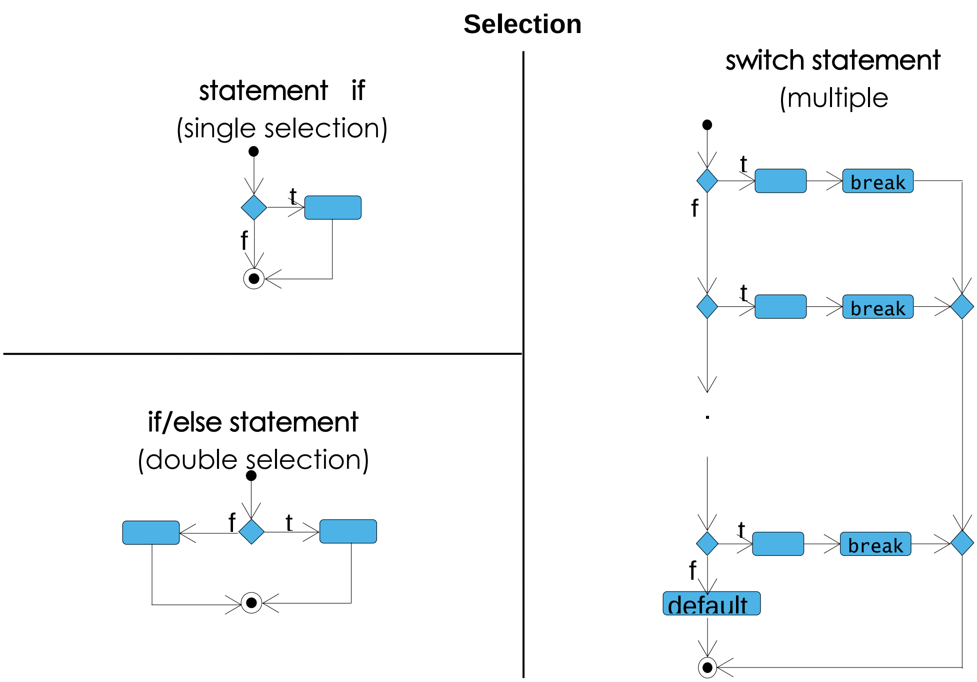
يمثل هذا النوع من البُنى عملية تنفيذ جزء معين من البرنامج إذا تحقق شرط ما.

بالنسبة لبنى الاختيار فإنه يتوفر منها في لغة Java ثلاثة أنواع سوف نقوم بعرضها في هذا الفصل وهي:

بنية الاختيار if (البنية وحيدة الاختيار Single-selection structure) التي تسمح بتنفيذ (أو اختيار) فعل ما إذا كان الشرط المرافق لها صحيحًا، وإلا فإنها تقوم بتجاوز ذلك الفعل، أي أنها تسمح باختيار أو تجاهل فعل ما.

وبنية الاختيار if/else (البنية مضاعفة الاختيار Double-selection structure) التي تسمح بتنفيذ فعل إذا كان الشرط المرافق صحيحًا أو بتنفيذ فعل آخر مختلف في الحالة المعاكسة لذلك، أي أنها تسمح بالاختيار ما بين فعلين مختلفين.

أمـا بنيـة الاختيـار switch (البنيـة متعـددة الاختيار structure (Multiple-selection فهـي تسمح بتنفيذ أحد الأفعال المختلفة الممكنة وذلك حسب قيمة عبارة ما، أي أنها تسمح باختيار فعل محدد من بين مجموعة ممكنة من الأفعال المختلفة.



المخططات الصندوقية لبنى الاختيار

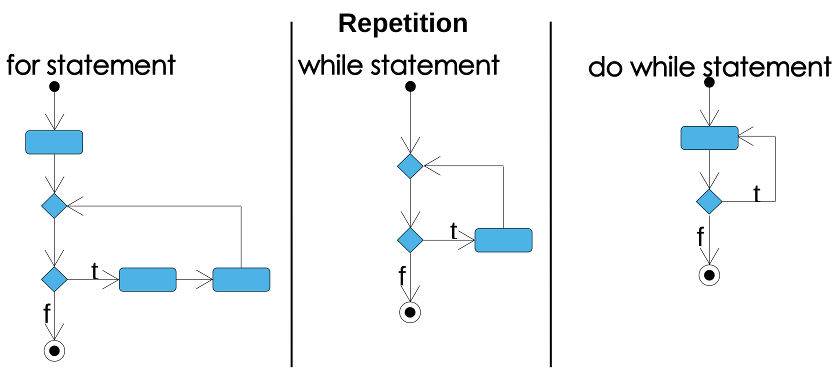
بنية التكرار Repetition Structure:

يمثل هذا النوع من البُنى عملية إعادة تنفيذ أجزاء من البرنامج عدد معين من المرات أو حتى يتحقق شرط معين. وتمتلك لغة Java ثلاث بنى تكرارية سوف نتطرق إليها بالتفصيل لاحقًا هي:

بنية التكرار for تستخدم هذه البُنية لتكرار تنفيذ فعل ما لعدد محدد من المرات.

وبنية التكرار while وتستخدم لتنفيذ فعل ما عدد غير محدود من المرات بشرط تحقق الشرط المرافق.

والبنية التكرارية do/while والتي تنفذ فعل ما عدد غير محدود من المرات إلى أن لا يتحقق الشرط المرافق.



المخططات الصندوقية للبنى التكرارية

وبهذا فإن لغة Java تحتوي على سبعة بُنى تحكم هي: البنية التسلسلية، ثلاث أنواع من بنى الاختيار وثلاثة أنواع من البنى التكرارية.

يتم بناء البرامج المكتوبة بلغة Java من خلال جمع عدد من بنى التحكم المناسبة لخوارزمية هذا البرنامج. كما لاحظنا في المخططات السابقة تم التعبير عن البنى بما يسمى المخططات الصندوقية وتبدأ بدائرة صغيرة وتنتهي بها.

تسهل هذه المخططات عملية بناء البرنامج حيث نقوم بوصل البنى بعضها ببعض عندما نصل نقطة النهاية لبنية ما بقطة البداية لبنية أخرى تالية لها. وتسمى هذه العملية بعملية تكديس بنى التحكم control-structure stacking.

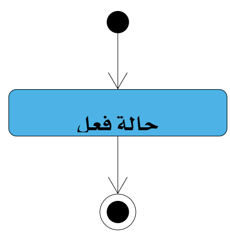
يوجد أيضًا طريقة أخرى إضافية للوصل بين بنى التحكم، هي طريقة تداخل بنى التحكم control-structure nesting.

أسلوب البرمجة المُهيكلة Structured Programming:

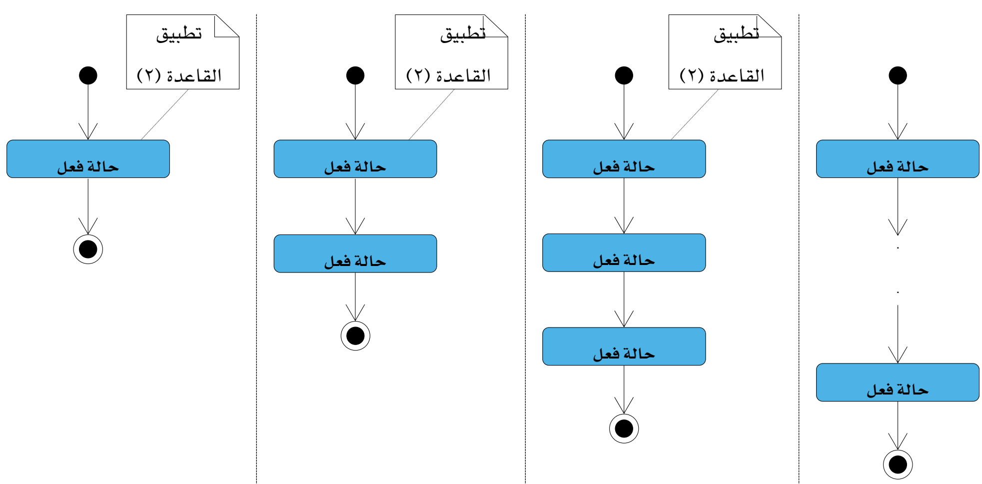
يجب على المبرمجين تصميم برامجهم بطريقة مهيكلة إذ تساعد هذه الطريقة على إنتاج برامج أسهل للفهم وللفحص ولكشف العيوب والتعديل أكثر من مثيلاتها غير المهيكلة. توجد قواعد يجب مراعاتها من أجل بناء برامج مُهيكلة.

قواعد تشكيل برامج مُهيكلة:

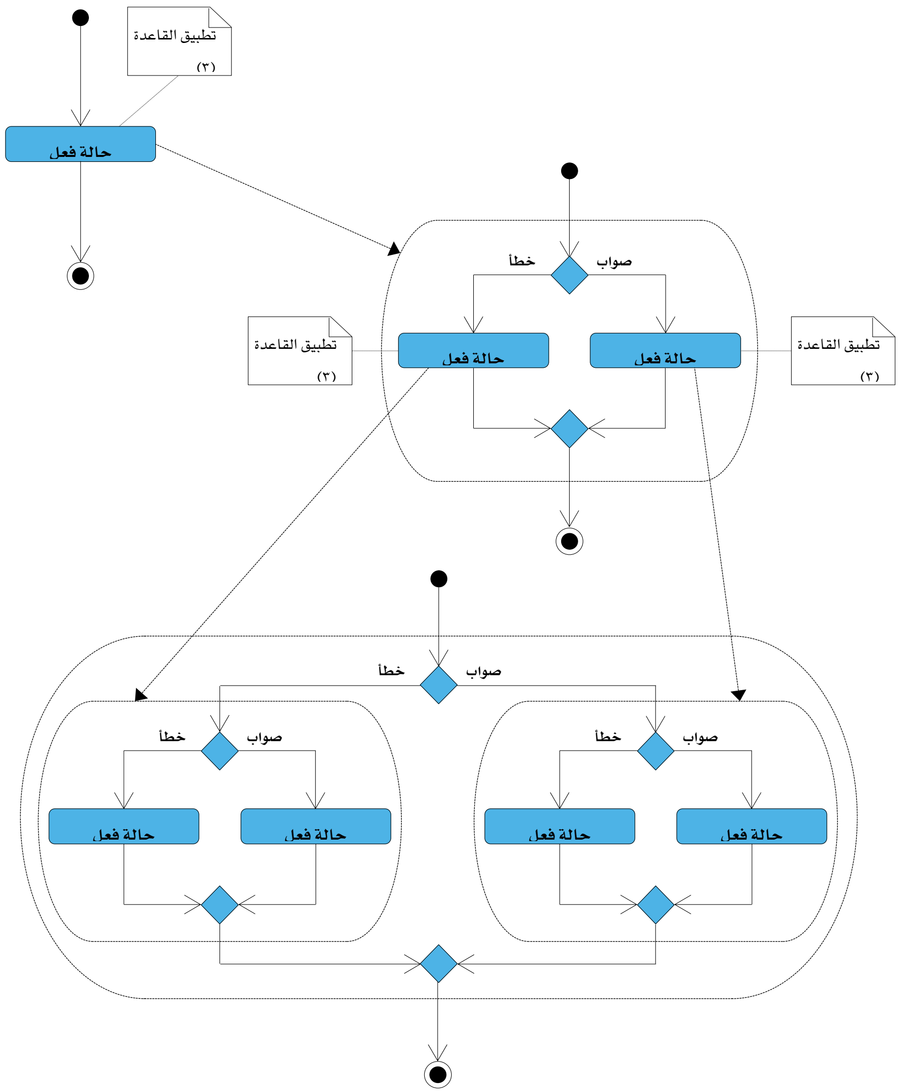
١ . ابدأ مع ابسط المخططات الصندوقية.



٢ . يمكن الاستعاضة عن أي فعل بفعلين موضوعين على التسلسل.



٣ . يمكن الاستعاضة عن أي فعل ببنية للتحكم (تسلسلية، if، if/else، switch، for، while، do/while):



٤ . يمكن تطبيق القاعدتين (٢) و(٣) قدر ما تشاء وفي أي ترتيب تريد.

بنية الاختيار if (البنية وحيدة الاختيار Single-selection structure):

يجري استخدام بنية الاختيار من أجل تحديد مسار فعل ما من بين مجموعة من المسارات المتاحة. فإذا كانت على سبيل المثال، درجة النجاح في مادة ما تساوي ٥٠ درجة، عندها فإن تعليمة لغة الخوارزميات التالية:

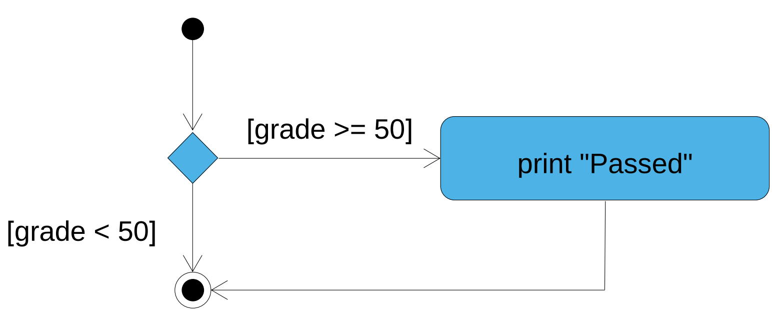
إذا كانت درجة الطالب أكبر من أو تساوي ٥٠ اطبع «ناجح»	if student's grad is greater than or equal to 50 print «Passed»

تقوم أولاً من التحقق من صحة أو عدم صحة الشرط الذي يفحص فيما إذا كانت درجة الطالب أكبر من أو تساوي القيمة ٥٠. ثم إذا كان الشرط محققاً عندها يتم طباعة العبارة المطلوبة «ناجح Passed» وإلا فيتم تنفيذ تعليمة لغة الخوارزمية التالية حسب تسلسل ترتيب التعليمات، أي أنه إذا لم يتحقق الشرط يتم تجاهل تعليمة الطباعة ليتم تنفيذ التعليمات التالية لها.

يمكن استبدال تعليمة لغة الخوارزميات السابقة بالتعليمة التالية بلغة Java:

```
if ( grad >= 50 )
      System.out.println( "Passed" );
```

والتي يمكن تمثيلها بالمخطط الصندوقي التالي:



تأخذ عبارة if كوسيط لها شرطًا من النمط boolean يكتب بين قوسين، بحيث يمكن أن ينتج عن هذا التعبير إحدى قيمتين true أو false، فإذا كانت قيمة التعبير هي false تتجاهل لغة Java التعليمة التالية أو كتلة التعليمات التالية، وتستخدم تعليمة if وفق الصيغة التالية:

```
if ( condition )
      statement;
```

أو:

```
if ( condition )
{
      statement_1;
      statement_2;
      ...
      statement_n;
}
```

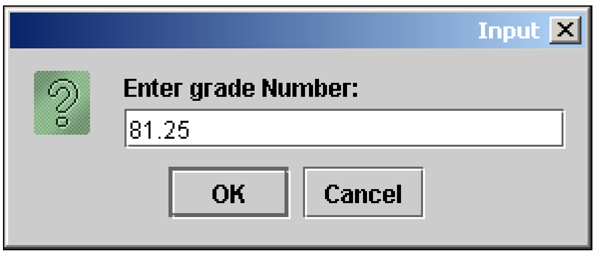
لجعل عبارة if تمتلك أكثر من عبارة برمجية لابد من حصر تلك العبارات بين القوسين {}.

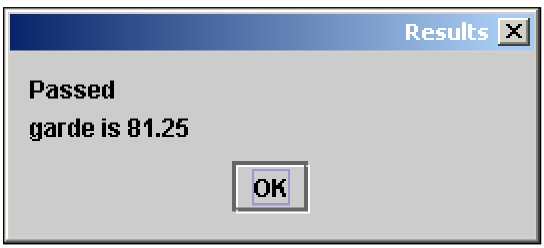
سنورد هنا برنامجًا يطلب من المستخدم إدخال درجته ومن ثم يقوم بطباعة ناجح في حال تحقق الشرط:

```
import javax.swing.JOptionPane;
public class IfTest
{
      public static void main( String args[] )
      {
      float grade;
      grade = Float.parseFloat(
                  JOptionPane.showInputDialog(
                                    "Enter grade Number:" ));
```

```
      if ( grade >= 50 )
           JOptionPane.showMessageDialog( null,
                " Passed \ngarde is " + grade ,
                "Results", JOptionPane.PLAIN_MESSAGE );

      System.exit( 0 );
      }
}
```





ملحوظة: ❖ يمكن أن يكون الشرط في عبارة if هو تعبير منطقي فمثلاً:

```
if ( i % 2 == 0 ) {…}
```

❖ كما يمكن أن يكون متغيرًا من النمط boolean:

```
boolean b = true;
if ( b ) {…}
```

❖ ينصح عادةً بكتابة الأقواس {} حتى وإن استخدمت عبارة واحدة للتعليمة if بحيث يكون البرنامج أكثر سهولة في القراءة والفهم والتعديل.

❖ يفضل وضع مسافة قبل العبارة التي تنتمي للتعليمة السابقة لها لتسهيل عملية الفهم والتعديل فمثلاً التعليمة:

```
if ( i % 2 == 0 )
     print( "bnvv" );
```

أفضل من التعليمة:

```
if (i % 2 == 0 )
print( "bnvv" );
```

بنية الاختيار if/else (البنية مضاعفة الاختيار Double-selection structure):

تسمح هذه البنية بتحديد جملة من الأفعال الممكن تنفيذها إذا كان الشرط المرافق صحيحًا أو إذا لم يكن كذلك. على سبيل المثال، تقوم تعليمة لغة الخوارزميات التالية:

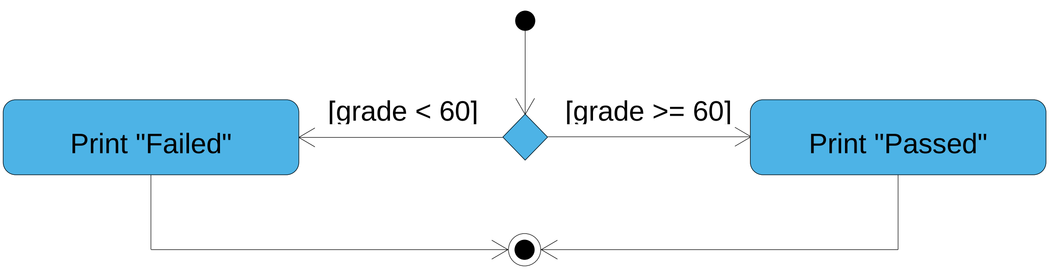
إذا كانت درجة الطالب أكبر من أو تساوي ٥٠	if student's grad is greater than or equal to 50
اطبع «ناجح»	print «Passed»
وإلا	Else
اطبع «راسب»	print «Failed»

بطباعة كلمة ناجح Passed إذا كانت درجة الطالب أكبر من أو تساوي القيمة ٥٠ درجة وإلا فهي تطبع الكلمة راسب Failed. وفي كلا الحالتين، ينتقل التنفيذ بعد القيام بعملية الطباعة إلى تعليمة لغة الخوارزميات التالية للتعليمة السابقة.

يمكن كتابة تعليمة لغة الخوارزميات السابقة بلغة Java على الشكل التالي:

```
if ( grad >= 50 )
      System.out.println( "Passed" );
else
      System.out.println( "Failed" );
```

وتمثل تعليمة if/else السابقة بمخطط النشاط التالي:



تستخدم تعليمة if/else وفق الصيغة التالية:

```
if ( condition )
      statement;
else
      statement;
```

أو:

```
if ( condition )
{... }
else
{
```

```
      statement_1;
      statement_2;
      ...
      statement_n;
}
```

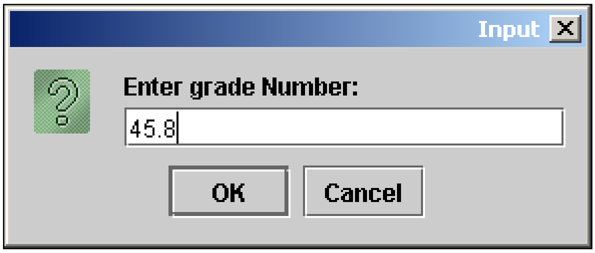
لجعل عبارة if/else تمتلك أكثر من عبارة برمجية لابد من حصر تلك العبارات بين القوسين {}، كما هو الحال مع التعليمة if.

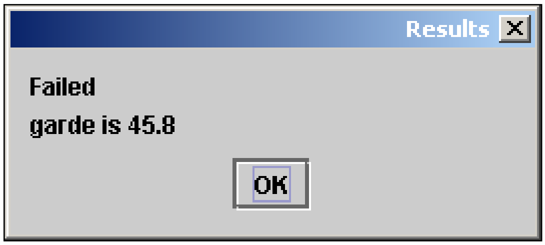
سنورد هنا برنامجًا يطلب من المستخدم إدخال درجته ومن ثم يقوم بطباعة ناجح أو راسب حسب القيمة:

```
import javax.swing.JOptionPane;
public class IfTest {
      public static void main( String args[] )
      {
            float grade;
            grade = Float.parseFloat(
                        JOptionPane.showInputDialog(
                        "Enter grade Number:" ));

            if ( grade >= 50 )
                  JOptionPane.showMessageDialog( null,
                        " Passed \ngarde is " + grade ,
                        "Results", JOptionPane.PLAIN_MESSAGE
                        );
            else
                  JOptionPane.showMessageDialog( null,
                        " Failed \ngarde is " + grade ,
                        "Results", JOptionPane.PLAIN_MESSAGE
                        );

      System.exit( 0 );
      }
}
```





المعامل الشرطي Conditional Operator:

يرمز لهذا العملية بالرمز (?:) وهي تشبه البنية if/else. وهي عملية ثلاثية تأخذ ثلاثة وسائط كالتالي:

```
(conditional-expression) ? expression-if-true: expression-
if-false
```

فمثلاً يمكننا كتابة تعليمة if/else التالية:

```
if ( grad >= 50 )
      System.out.println( "Passed" );
else
      System.out.println( "Failed" );
```

بالشكل التالي:

```
System.out.println( ( grad >= 50 ) ? "Passed": "Failed" );
```

والتعليمة:

```
if ( y < 5 )
```

```
      x = 4 * y;
else
      x = 2 * y;
```

بالشكل:

```
x = ( y < 5 ) ? 4 * y: 2 * y;
```

التعليمات الشرطية المتداخلة Nested if/else Statements:

في حالة تعددت الاحتمالات وزادت الحالات يفضل استخدام عملية تداخل التعليمات الشرطية. فمثلاً:

```
if ( grad >= 90 )
      System.out.println( "Excellent" );
else if ( grad >= 80 )
      System.out.println( "Very Good" );
else if ( grad >= 65 )
      System.out.println( "Good" );
else if ( grad >= 50 )
      System.out.println( "Pass" );
else
      System.out.println( "Fail" );
```

بنية الاختيار switch (البنية متعددة الاختيار Multiple-selection structure):

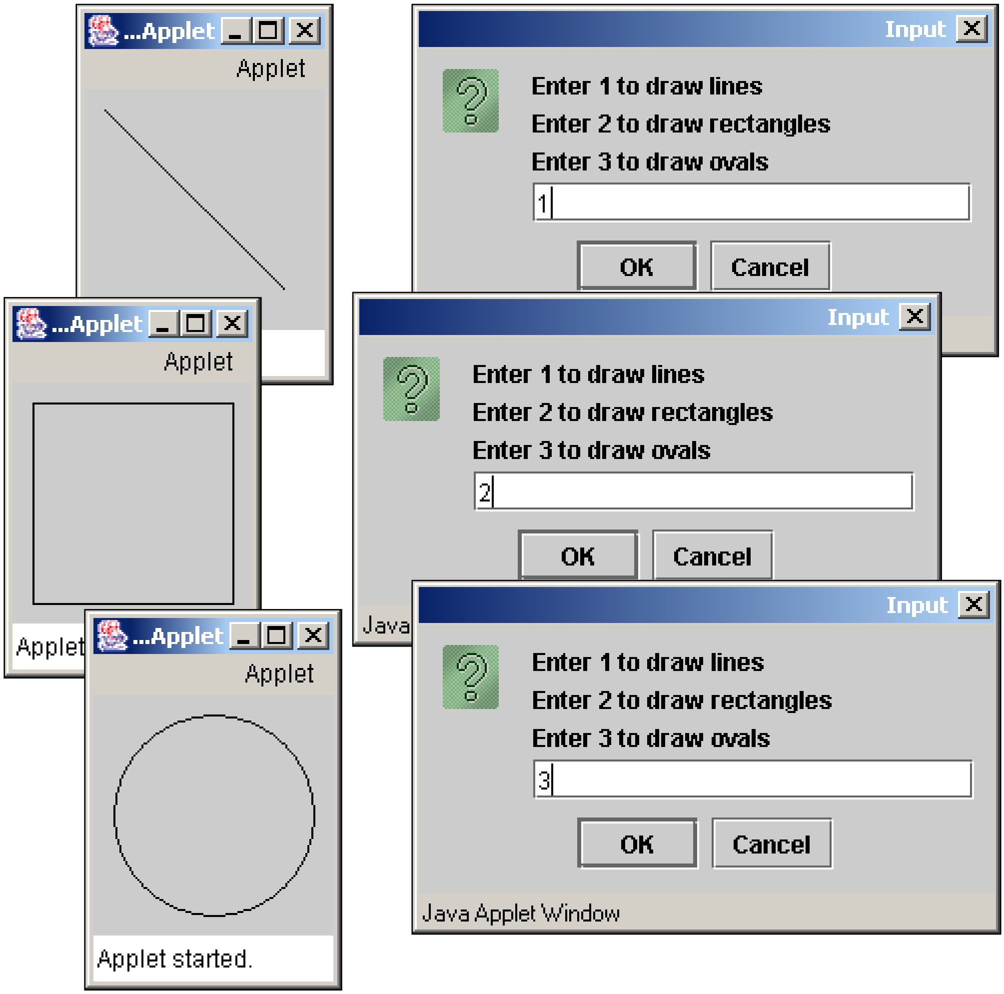
تتألف هذه البنية من سلسلة من الأجزاء التي تبدأ بكلمة case ومن جزء اختياري هو الجزء default. المثال التالي لتوضيح عملية استخدام التعليمة switch:

```
import java.awt.Graphics;
import javax.swing.*;
```

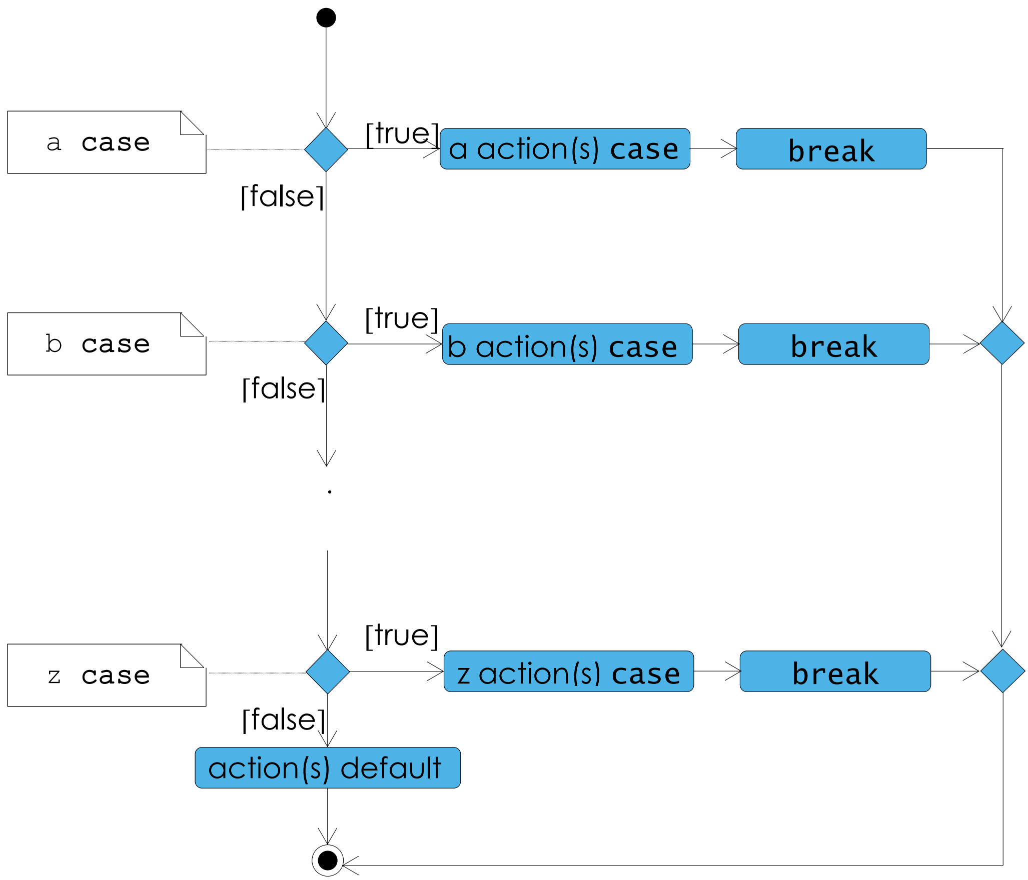
```
public class SwitchTest extends JApplet {
      int choice; // user's choice of which shape to draw
            // initialize applet by obtaining user's choice

      public void init()
      {
            choice = Integer.parseInt(
                        JOptionPane.showInputDialog(
                        "Enter 1 to draw lines\n" +
                        "Enter 2 to draw rectangles\n" +
                        "Enter 3 to draw ovals\n" ));
      } // end method init
         // draw shapes on applet's background

      public void paint( Graphics g )
      {
            switch ( choice ) { // determine shape to draw
                  case 1: // draw a line
                        g.drawLine( 10, 10, 100, 100 );
                        break; // done processing case
                  case 2: // draw a rectangle
                        g.drawRect( 10, 10, 100, 100 );
                        break; // done processing case
                  case 3: // draw an oval
                        g.drawOval( 10, 10, 100, 100 );
                        break; // done processing case
                  default:   // draw string indicating
                              //invalid value entered
                        g.drawString(
                              "Invalid value entered",
                              10, 40 );
            } // end switch
      } // end method paint
} // end class SwitchTest
```



يمكن تمثيل البنية switch ذات الخيارات المتعددة بالشكل التالي:



بنية التكرار for (حلقة التكرار ذات العداد):

تستخدم البنية for لتكرار مجموعة من التعليمات عدد من المرات. يمكن صياغة الشكل العام للبنية for على الشكل التالي:

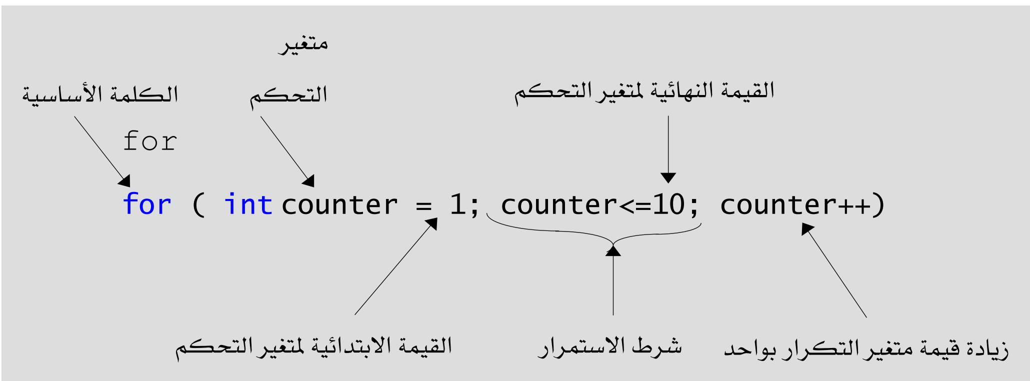
```
for(initialization; loopContinuationCondition; increment)
      statement;
```

أو:

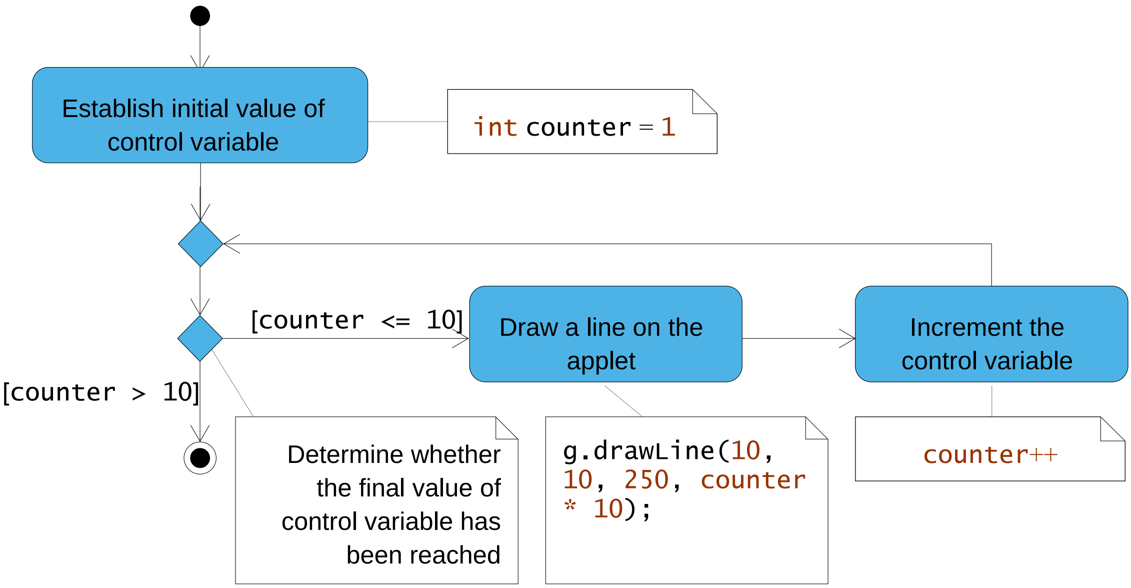
```
for(initialization; loopContinuationCondition; increment)
{
```

```
        statement_1;
        statement_2;
        ...
        statement_n;
}
```

حيث تقوم initialization بإعطاء قيمة ابتدائية لمتغير التحكم بالتكرار وهي عبارة تنفذ مرة واحدة فقط قبل إجراء عملية التكرار، ويشكل loopContinuationCondition شرط الاستمرار في التكرار، أما increment فيقوم بزيادة قيمة متغير التحكم بالتكرار.



ويمثل المخطط التالي مخطط النشاط لبنية for نموذجية:



وهذه مجموعة من الأمثلة توضح عملية التكرار:

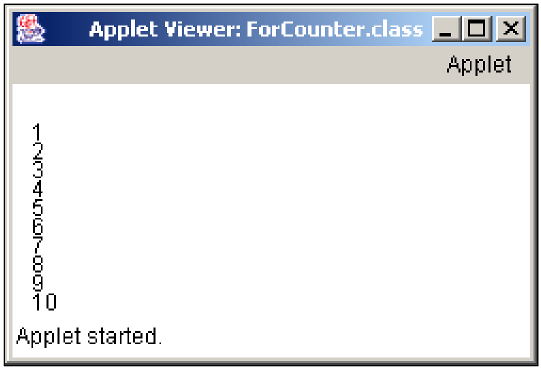
```
import java.awt.Graphics;
import javax.swing.JApplet;

public class ForCounter extends JApplet {
      public void paint( Graphics g )
      {
            String output="";

            for(int counter = 1; counter <= 10; counter++ )
            {
                  output += counter؛

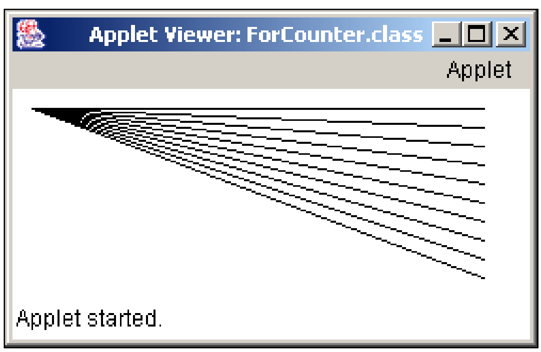
                  g.drawString( output, 10,
                        20 + counter * 10 );

                  output="";
            }
      }
}
```




```
import java.awt.Graphics;
import javax.swing.JApplet;

public class ForCounter extends JApplet {
      public void paint( Graphics g )
      {
            for(int counter = 1; counter <= 10; counter++ )
                  g.drawLine( 10, 10, 250, counter * 10 );
      }
}
```



وهذا برنامج يستخدم لحساب المضروب لعدد يعطيه المستخدم:

```
import javax.swing.JOptionPane;

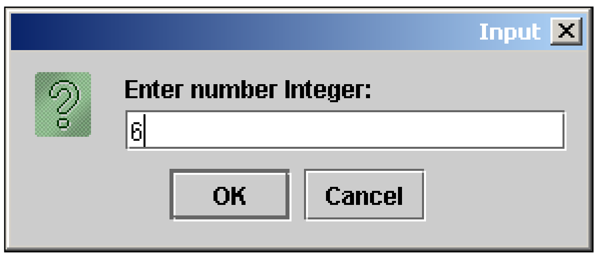
public class Factorial {
      public static void main( String[] args )
```

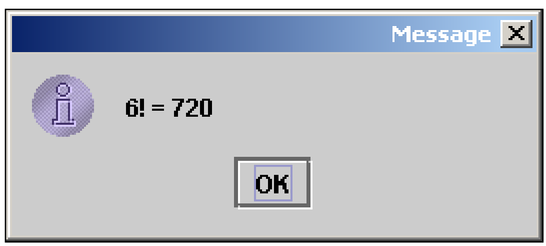
```
        {
                int fact=1, n;
                n=Integer.parseInt(JOptionPane.showInputDialog(
                                "Enter number Integer:"));

                for ( int i = 1; i <= n; i++ )
                        fact *= i;

                JOptionPane.showMessageDialog( null,
                        n + "! = " + fact );

                System.exit( 0 );
        }
}
```





وهذا البرنامج يقوم برسم شكل مثلث:

```
import javax.swing.JOptionPane;

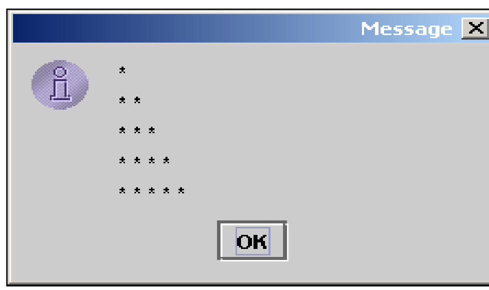
public class Triangle1 {
        public static void main( String[] args ){
                int n = 5, i, j;
                String output = "";
```

```
        for ( i = 1; i <= n; i++ )
        {
            for ( j = 1; j <= i; j++ )
                output += "* ";

            output += "\n";
        }

        JOptionPane.showMessageDialog( null, output );

        System.exit( 0 );
    }
}
```




```
public class Triangle2 {
    public static void main( String[] args )
    {
        int n = 5, i, j;
        String output = "";

        for ( i = 1; i <= n; i++ )
        {
            for ( j = 1; j <= n - i; j++ )
                output += " ";

            for ( j = 1; j <= i; j++ )
                output += "*";

            output += "\n";
        }
```

```
            System.out.println( output );
      }
}
```

أما في هذا البرنامج فيكون الخرج كالتالي:

```
    *
   **
  ***
 ****
*****
```

البرنامج التالي يقوم بطباعة الأعداد الزوجية المحصورة بين ٠ و١٠٠ والتي **لا تقبل** القسمة على ٣ بدون باقي:

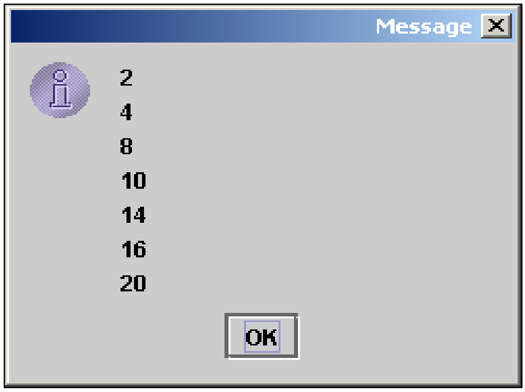
```
import javax.swing.JOptionPane;

public class math1 {
      public static void main( String[] args )
      {
            String output = "";

            for ( int i = 0; i <= 20; i++ )
                  if ( i % 2 ==0 && i % 3 != 0 )
                        output += i + "\n";

            JOptionPane.showMessageDialog( null, output );

      System.exit( 0 );
      }
}
```



برنامج لحساب مجموع الأعداد الزوجية المحصورة بين ٢ و١٠٠:

```
// Summing integers with the for statement.
import javax.swing.JOptionPane;

public class Sum {
      public static void main( String args[] )
      {
            int total = 0;
            for(int number = 2; number <= 100; number += 2)
                  total += number;

            JOptionPane.showMessageDialog( null,
                  "The sum is " + total,
                  "Total Even Integers from 2 to 100",
                  JOptionPane.INFORMATION_MESSAGE );

      System.exit( 0 );
      }
}
```



برنامج يقوم بحساب قيمة المجموع التالي:

$$sum = \sum_{i=1}^{10} \frac{2^i - 5}{i}$$

```
import javax.swing.JOptionPane;

public class mySum {
      public static void main( String[] args )
      {
            double sum = 0, pow2i;
            int i, j;

            for (i = 1; i <= 10; i++ ) {
                  pow2i = 1;

                  for (j = 1; j <= i; j++ )
                        pow2i *= 2;

                  sum += (pow2i - 5)/i;
            }

            JOptionPane.showMessageDialog( null,
                  "sum = " + sum );

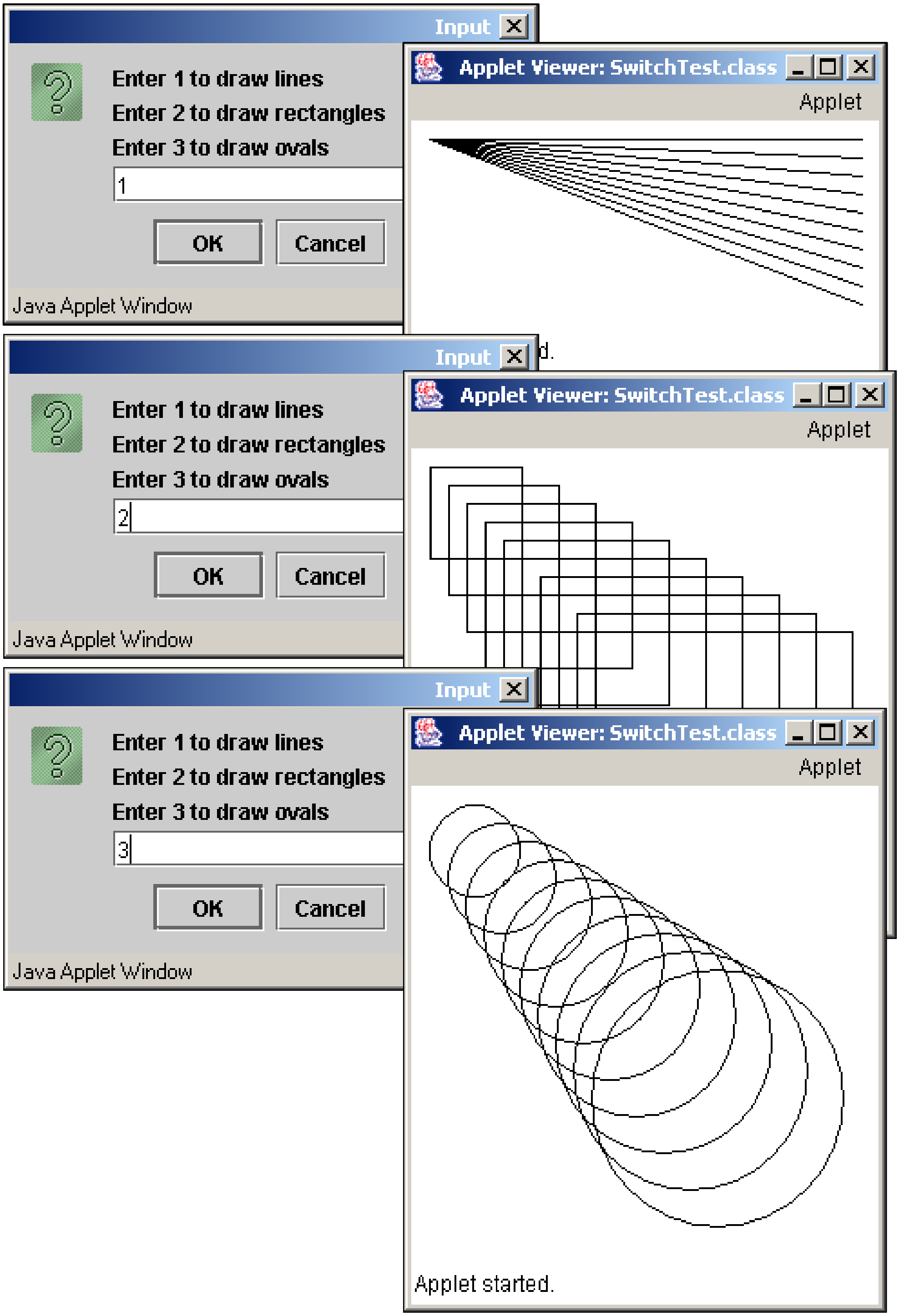
      System.exit( 0 );
      }
}
```

تعديل على برنامج اختبار العبارة switch السابق الذكر:

```
import java.awt.Graphics;
import javax.swing.*;


public class SwitchTest extends JApplet {
      int choice;
```

```
	public void init()
	{
		String input;
		input = JOptionPane.showInputDialog(
			"Enter 1 to draw lines\n" +
			"Enter 2 to draw rectangles\n" +
			"Enter 3 to draw ovals\n" );
		choice = Integer.parseInt( input );
	}
	public void paint( Graphics g )
	{
		for ( int i = 0; i < 10; i++ ) {
			switch ( choice ) {
				case 1:
					g.drawLine( 10, 10, 250,
						10 + i * 10 );
					break;
				case 2:
					g.drawRect( 10 + i * 10,
						10 + i * 10, 50 + i * 10,
						50 + i * 10 );
					break;
				case 3:
					g.drawOval( 10 + i * 10,
						10 + i * 10, 50 + i * 10,
						50 + i * 10 );
					break;
				default:
					g.drawString(
						"Invalid value entered",
						10, 20 + i * 15 );
			}
		}
	}
}
```


Input
Enter 1 to draw lines
Enter 2 to draw rectangles
Enter 3 to draw ovals
1
OK
Cancel
Java Applet Window
Applet Viewer: SwitchTest.class
Applet
Input
Enter 1 to draw lines
Enter 2 to draw rectangles
Enter 3 to draw ovals
2
OK
Cancel
Java Applet Window
Applet Viewer: SwitchTest.class
Applet
Input
Enter 1 to draw lines
Enter 2 to draw rectangles
Enter 3 to draw ovals
3
OK
Cancel
Java Applet Window
Applet Viewer: SwitchTest.class
Applet
Applet started.

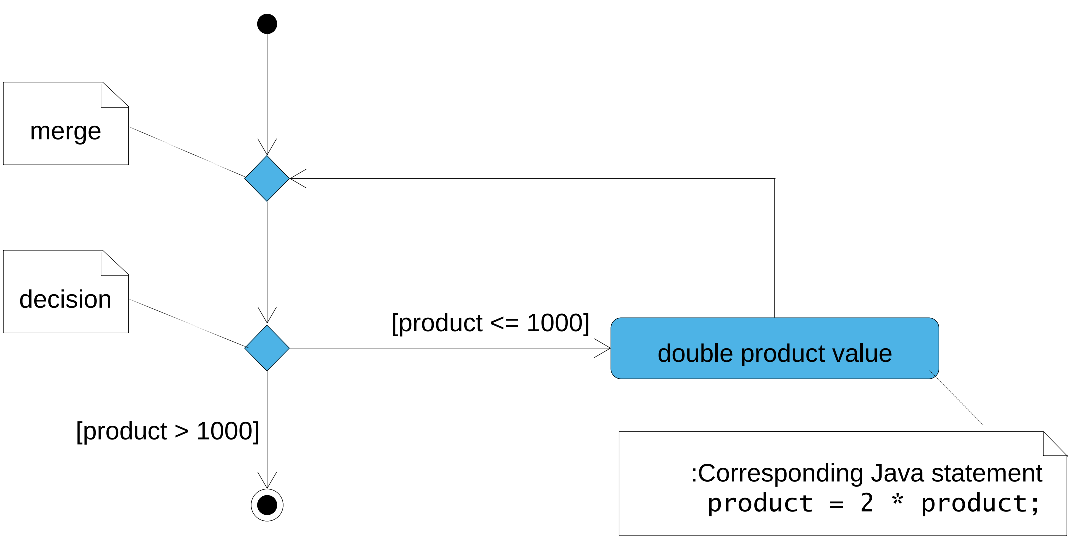
بنية التكرار while:

تسمح هذه البنية التكرارية بتحديد مجموعة من الأفعال يجري تكرارها طالما ظلّ الشرط المرافق للبنية محققاً. ويكون مخطط هذه البنية كالتالي:

لنأخذ كمثال يوضح استخدام while، جزءًا من برنامج يقوم بتحديد أول قيمة من الشكل ٢^{n} (n>=1) أكبر من ١٠٠٠، بفرض أن p هو متغير صحيح يحتوي على القيمة ٢ كقيمة ابتدائية. عند نهاية البنية التكرارية while التالية سوف نحصل على القيمة التي نبحث عنها:

```
int p = 2;
while ( p <= 1000)
      p = 2 * p;
```

تتزايد قيمة المتغير p إلى أن تكون ١٠٢٤ وعندها يكون الشرط p <= 1000 غير متحققاً وعندها يتوقف التكرار.



بنية while ذات العداد:

كما علمنا من السابق فإن التكرار ذي العداد يستخدم للتكرار بعدد معين من المرات كما هو الحال في العبارة for ويمكننا كتابة عبارة while ذات عداد بنفس كيفية عبارة for. فمثلاً عبارة for التالية:

```
for(initialization; loopContinuationCondition; increment )
      statement;
```

يمكن كتابتها بالشكل التالي في عبارة while:

```
initialization;
while ( loopContinuationCondition ) {
      statement;
      increment;
}
```

والبرنامج التالي لتوضيح استخدام العبارة while ذات العداد. برنامج لحساب معدل درجات ١٠ طلاب في صف:

```
// Class-average program with
// counter-controlled repetition.

import javax.swing.JOptionPane;

public class Average1 {
      public static void main( String args[] )
      {
            int total; // sum of grades input by user
            int gradeCounter; // number of grade to
                              // be entered next
            int grade; // grade value
            int average; // average of grades
            String gradeString; // grade typed by user
```

```
            // initialization phase
            total = 0; // initialize total
            gradeCounter = 1; // initialize loop counter
                                  // processing phase
            while ( gradeCounter <= 10 ) { // loop 10 times
            // prompt for input and read grade from user
                  gradeString = JOptionPane.showInputDialog(
                        "Enter integer grade: " );

                  // convert gradeString to int
                  grade = Integer.parseInt( gradeString );

                  // add grade to total
                  total = total + grade;

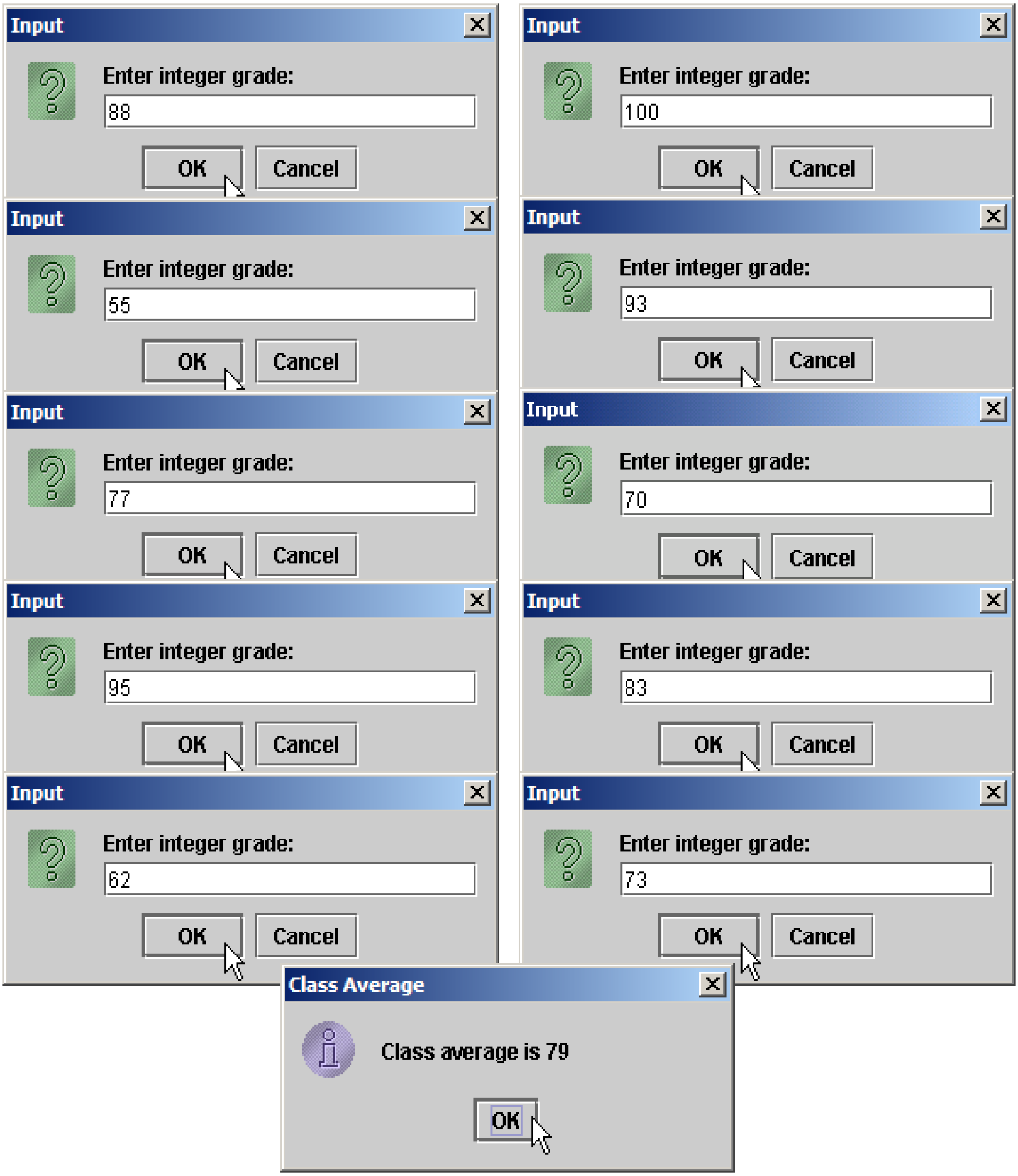
                  // increment counter
                  gradeCounter = gradeCounter + 1;

            } // end while

            // termination phase
            average = total / 10; // integer division

            // display average of exam grades
            JOptionPane.showMessageDialog( null,
                  "Class average is " + average,
                  "Class Average",
                  JOptionPane.INFORMATION_MESSAGE );

            System.exit( 0 ); // terminate the program
      } // end main
} // end class Average1
```


Input
Enter integer grade:
88
OK
Cancel
Input
Enter integer grade:
100
OK
Cancel
Input
Enter integer grade:
55
OK
Cancel
Input
Enter integer grade:
93
OK
Cancel
Input
Enter integer grade:
77
OK
Cancel
Input
Enter integer grade:
70
OK
Cancel
Input
Enter integer grade:
95
OK
Cancel
Input
Enter integer grade:
83
OK
Cancel
Input
Enter integer grade:
62
OK
Cancel
Input
Enter integer grade:
73
OK
Cancel
Class Average
Class average is 79
OK

بنية while بدون عداد:

والبرنـامج التـالي لتوضـيح اسـتخدام العبـارة while ذات قيمـة شـرطية. برنـامج لحسـاب معدل درجات طلاب صف:

```
// Class-average program with
// sentinel-controlled repetition.

import java.text.DecimalFormat; // class to format numbers
import javax.swing.JOptionPane;

public class Average2 {
      public static void main( String args[] )
      {
            int total;
            int gradeCounter;
            int grade;
            double average;
            String gradeString;

            total = 0;
            gradeCounter = 0;

            gradeString = JOptionPane.showInputDialog(
                  "Enter Integer Grade or -1 to Quit:" );
            grade = Integer.parseInt( gradeString );

            while ( grade != -1 ) {
                  total = total + grade;

                  gradeCounter = gradeCounter + 1;

                  gradeString = JOptionPane.showInputDialog(
                        "Enter Integer Grade or -1 to Quit:"
                        );

                  grade = Integer.parseInt( gradeString );
```

```
        }

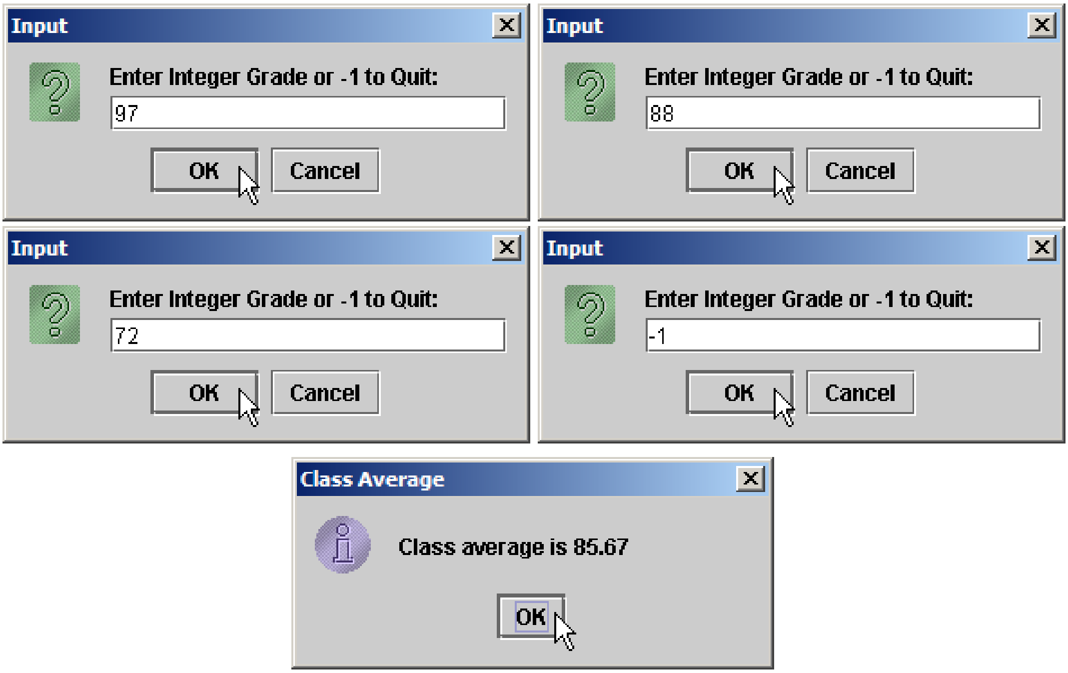
        DecimalFormat twoDigits = new DecimalFormat(
              "0.00" );

        if ( gradeCounter != 0 ) {

              average = (double) total / gradeCounter;

              JOptionPane.showMessageDialog( null,
                    "Class average is " +
                    twoDigits.format( average ),
                    "Class Average",
                    JOptionPane.INFORMATION_MESSAGE );
        }
        else
              JOptionPane.showMessageDialog( null,
                    "No grades were entered",
                    "Class Average",
                    JOptionPane.INFORMATION_MESSAGE );

        System.exit( 0 );
     }
}
```



بنية التكرار do/while:

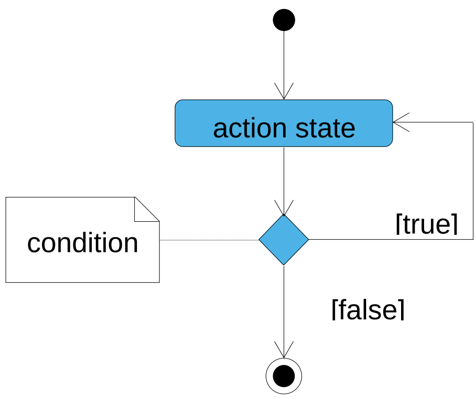
تشبه بنية do/while البنية while حيث نقوم بالبنية while بالتحقق من صحة شرط الاستمرار بالتكرار في بداية الحلقة قبل تنفيذها. أما في حالة البنية do/while فيتم ذلك بعد تنفيذ جسم الحلقة أولاً. أي يتم تنفيذ جسم البنية do/while مرة واحدة على الأقل. وتكون الصيغة العامة للعبارة do/while بالشكل:

```
do
     statement;
while ( condition );
```

أو:

```
do {
      statement_1;
      statement_2;
      ...
      statement_n;
} while ( condition );
```

ومخطط النشاط للبنية do/while كالتالي:



البريمج التالي يوضح استخدام العبارة do/while:

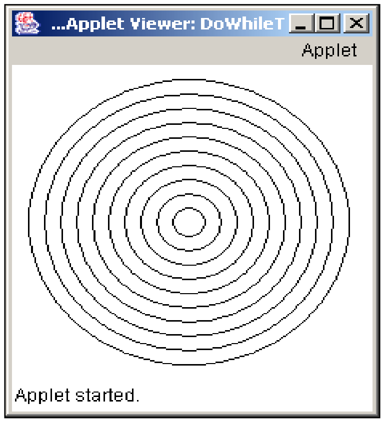
```
// Using the do...while statement.

import java.awt.Graphics;
import javax.swing.JApplet;

public class DoWhileTest extends JApplet {
      // draw lines on applet
      public void paint( Graphics g )
      {
            int counter = 1; // initialize counter
            do {
                  g.drawOval( 110 - counter * 10,
                        110 - counter * 10,
                        counter * 20, counter * 20 );
```

```
                 ++counter;
           } while ( counter <= 10 ); // end do...while

      } // end method paint
} // end class DoWhileTest
```



التعليمتان break و continue:

تستخدم التعليمتان break و continue من أجل تغيير مجرى تدفق التحكم ضمن البرنامج.

تسبب التعليمة break عند تنفيذها مع أحد البنى switch, for, while, do/while الخروج المباشر من هذه البنية ويتابع البرنامج بعدها تنفيذه مع أول تعليمة تلي البنية مباشرة. البرنامج التالي يبين استخدام التعليمة break:

```
// Terminating a loop with break.

import javax.swing.JOptionPane;

public class BreakTest {
     public static void main( String args[] )
     {
```

```
            String output = "";
            int count;

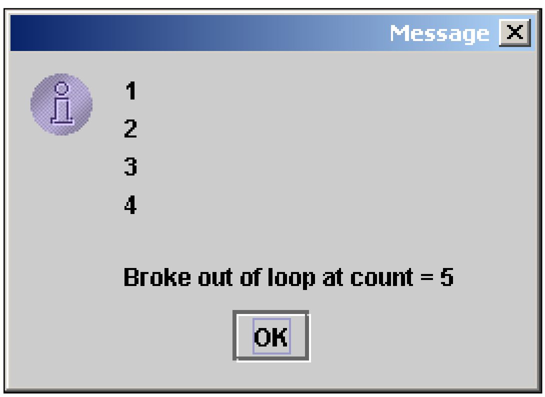
            // loop 10 times
            for ( count = 1; count <= 10; count++ ) {
                 if ( count == 5 ) // if count is 5,
                      break; // terminate loop

                 output += count + "\n";
            } // end for

            output += "\nBroke out of loop at count = " +
                      count;

            JOptionPane.showMessageDialog( null, output );

            System.exit( 0 ); // terminate application
       } // end main
} // end class BreakTest
```



تسـبب التعليمـة continue عنـد تنفيـذها مـع أحـد البنـى do/while, while, for تجـاوز مـا تبقـى مـن التعليمـات في جسـم التكـرار والمتابعـة مـع المـرور التـالي في الحلقـة. يـتم في البنيـتين do/while, while القيـام بـالتحقق مـن صـحة شـرط اسـتمرار التكـرار مباشـرةً بعـد تنفيـذ التعليمـة continue. أمـا مـع البنيـة for، فيـتم القيـام بعمليـة الزيـادة

أولاً ثـم يبـدأ بعـدها تقيـيم شـرط اسـتمرار التكـرار. البرنـامج التـالي يـبين اسـتخدام التعليمة continue:

```
// Continuing with the next iteration of a loop.

import javax.swing.JOptionPane;

public class ContinueTest {
      public static void main( String args[] )
      {
            String output = "";

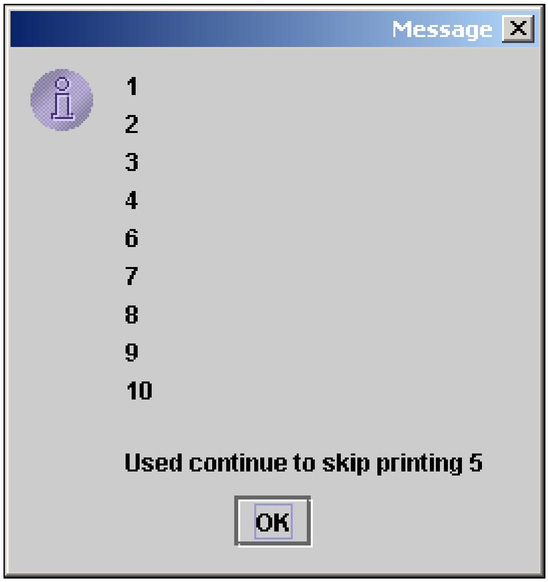
            for ( int count = 1; count <= 10; count++ ) {
            // loop 10 times
                  if ( count == 5 ) // if count is 5,
                        continue;
                        // skip remaining code in loop

                  output += count + "\n";
            } // end for

            output += "\nUsed continue to skip printing 5";

            JOptionPane.showMessageDialog( null, output );

            System.exit( 0 ); // terminate application
      } // end main
} // end class ContinueTest
```



العمليات المنطقية (العبارات المنطقية المركبة):

لتركيب عبارتين أو أكثر من العبارات المنطقية يجب استخدام العمليات المنطقية operators logical التي تمكننا من تشكيل شروط معقدة من خلال الجمع بين عدة شروط بسيطة. هذه العمليات هي: **&&** (عملية «و And» المنطقية) و **||** (عملية «أو Or» المنطقية) و **!** (عملية «ليس not» المنطقية أو ما يسمى عملية النفي). يمكن دمج أي عبارتين منطقيتين بالشكل التالي:

```
expression && expression
expression || expression
! expression
```

أو

```
( expression ) && ( expression )
( expression ) || ( expression )
! ( expression )
```

جدول الحقيقة الخاص بالعملية المنطقية && «و And» المنطقية:

expression1	expression2	expression1 && expression2
true	true	true
false	true	false
true	false	false
false	false	false

يمكن تلخيص هذا الجدول بالعبارة التالية: «تكون العبارة المركبة بالعملية **&&** صحيحة فقط في حالة كانت كافة العبارات صحيحة وتكون خاطئة في حالة كانت إحدى العبارات خاطئة».

جدول الحقيقة الخاص بالعملية المنطقية || «أو Or» المنطقية:

expression1	expression2	expression1 || expression2
true	true	true
false	true	true
true	false	true
false	false	false

يمكن تلخيص هذا الجدول بالعبارة التالية: «تكون العبارة المركبة بالعملية **||** صحيحة في حالة كانت إحدى العبارات صحيحة وتكون خاطئة في حالة كانت كافة العبارات خاطئة».

جدول الحقيقة الخاص بالعملية المنطقية ! «ليس not» المنطقية:

expression	! expression
true	false
false	true

تمارين الفصل الرابع:

١ . ما نعني بالخوارزميات، وما هي أهم فوائدها؟

٢ . ما المقصود ببنية التكرار، لماذا تستخدم؟

٣. املأ الفراغات التالية:

- يمكن كتابة جميع البرامج باستخدام ثلاثة أنواع من بنى التحكم:.................. و.................. و..................
- تُستخدم البنية.................. من أجل تنفيذ فعل ما عندما يكون شرطها محققاً وتقوم بتنفيذ فعل آخر عندما يكون ذلك الشرط غير محققاً.
- تسمى عملية تكرار مجموعة من التعليمات عدداً من المرات بعملية التكرار ذات..................

٤ . ما هي نتيجة تنفيذ البرنامج التالي:

```
public class Printing{
    public static void main( String [] args ) {
        for ( int i = 1; i <= 10; i++) {
            for ( int j = 1; j <= 5; j++)
                System.out.print( '@' );
            System.out.println();
        }
    }
}
```

٥ . قم بكتابة برنامج يقوم بإيجاد جذري المعادلة: $\mathbf{a}x^2+\mathbf{b}x+\mathbf{c}=0$

حسب القانون: $x=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$.

٦ . قم بكتابة برنامج يقوم بطباعة كافة الأعداد الأولية المحصورة بين العددين ٠ و١٠٠. (العدد الأولي هو العدد الذي لا يقبل القسمة بدون باقي إلا على نفسه وعلى الرقم ١).

٧ . قم بكتابة برنامج يقوم بحساب مجموع الأعداد الصحيحة من ١ حتى ١٠. استخدم البنية while.

٨ . قم بكتابة برنامج يقوم بحساب x^y (x مرفوعة للقوة y). استخدم العبارة do/while (علمًا بأن x^y تعني أن العدد x تم ضربه في نفسه y من المرات).

٩ . اكتب برنامجًا يقوم بحساب القيمة e^x باستخدام الصيغة:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

١٠ . اكتب برنامجًا يقوم بحساب قيمة العدد π وذلك باستخدام السلسلة التالية: $\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \ldots$

١١ . اكتب برنامجا يقوم بطباعة النماذج التالية واحدًا تلو الآخر:

```
(A)            (B)                 (C)
*********      *********           *
********        ********          ***
*******          *******         *****
******            ******        *******
*****              *****       *********
****                ****        *******
***                  ***         *****
**                    **          ***
*                      *           *
```

الفصل الخامس

الطرق (التوابع)

Methods (Functions)

مقدمة:

إن معظم البرامج التي تتناول مسائل لها علاقة بالعالم الحقيقي المحيط بنا، هي برامج أكبر من تلك التي سوف نقوم بعرضها. ولقد برهنت التجربة على أن أفضل طريقة لتطوير وصيانة برنامج كبير، تتمثل في عملية تجزئته إلى قطع وأجزاء أصغر modules يمكن التحكم بها بسهولة أكبر من البرنامج الأصلي. تسمى هذه الطريقة بطريقة «فرّق تسد divide and conquer». يعرض هذا الفصل عدة إمكانيات للغة Java مساعدة في عملية تطوير وتصميم وتشغيل وصيانة البرامج الكبيرة.

أجزاء برنامج بلغة جافا (Program Modules in Java):

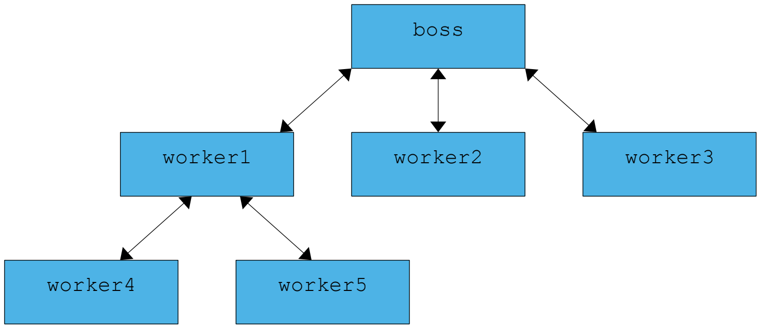
تمتلك البرامج في Java نوعين من الأجزاء الطرق methods والفئات classes. يتم كتابة برامج Java عادة بأن يتم الجمع بين الطرق المكتوبة من قبل المبرمج والطرق الجاهزة المتوفرة في المكتبات المعيارية للغة Java، كذلك هو الحال مع الفئات.

تحتوي مكتبات الطرق المعيارية في Java (Java Application Programming Interface «Java API» or Java class library) على مجموعة غنية من الطرق الخاصة بتنفيذ العمليات الحسابية الرياضية الشائعة، والعمليات الخاصة بسلاسل الحروف

بالإضافة إلى عمليات الدخل/الخرج وعمليات التحقق وكشف الأخطاء والكثير الكثير غيرها.

يستطيع المبرمج كتابة طرق للقيام بمهام محددة يمكن استخدامها في عدة نقاط ضمن نص البرنامج. تسمى هذه الطرق أحياناً «بطرق المبرمج Programmer-Defined Methods» حيث تجري كتابة التعليمات المعرفة للطريقة مرة واحدة فقط وتبقى بعدها هذه التعليمات معزولة ومخبئة بالنسبة لبقية الطرق.

يتم استدعاء أي طريقة من الطرق (التي كتبناها من أجل إنجاز المهمة الخاصة بها) من خلال استدعائها method Call. تتضمن عملية الاستدعاء اسم الطريقة المستدعاة والمعلومات اللازمة لها للقيام بعملها (على شكل وسطاء arguments). تشبه هذه العملية في البرمجة عملية الإدارة ذات التسلسل الهرمي حيث يطلب رب العمل (الذي يقوم باستدعاء الطرق) من عامله (الطريقة method) القيام بمهمة محددة ثم يعيد return النتائج التي حصل عليها. لا تعرف الطريقة المستدعية (رب العمل) كيف ستقوم الطريقة المستدعاة (العامل) بتنفيذ المهمة المناطة بها. يمكن للطريقة المستدعاة أن تستدعي طرق أخرى (عمال آخرين) دون أن تهتم الطريقة الأساسية المستدعية (رب العمل) لذلك.



علاقة التسلسل الهرمي بين الطريقة "رب العمل" والطريقة "العامل"

فئة (مكتبة) الطرق الرياضية Math-Class Methods:

تسمح طرق الفئة الرياضية للمبرمج بالقيام بالكثير من الحسابات الرياضية الشائعة. سنقوم باستخدام عدداً من الطرق الرياضية لتوضيح المفاهيم المتعلقة بالطرق وكيفية استخدامها.

يتم استدعاء الطرق عادةً بأن نكتب اسمها متبوعًا بقوس يساري مفتوح ووسيط (أو قائمة من الوسطاء يُفصل فيما بينها بواسطة فواصل« , ») ثم نضع أخيرًا قوس يميني. على سبيل المثال، يستطيع مبرمج يرغب في حساب وطباعة قيمة الجذر التربيعي للقيمة ٣، أن يكتب التعليمة التالية:

```
System.out.println( Math.sqrt( 3.0 ) );
```

يمكن لوسطاء الطريقة أن تكون ثوابت، متغيرات، أو حتى تعابير حسابية.

لنفرض مايلي:

```
float a = 1.0, b = 8.6, c = 42.6;
System.out.println( Math.sqrt( a + b * c ) );
```

الطرق الجاهزة:

سنورد في الجدول التالي بعضًا من طرق المكتبة الرياضية، حيث x وy هما من النمط double:

الطريقة	الوصف	مثال
abs(x)	طريقة القيمة المطلقة لـ x (هذه الطريقة تقبل أيضًا وسائط من النمط float, int, long)	abs(23.7) is 23.7 abs(0.0) is 0.0 abs(-23.7) is 23.7
ceil(x)	تقريب x إلى أصغر عدد صحيح	ceil(9.2) is 10.0 ceil(-9.8) is -9.0
cos(x)	طريقة جيب التمام لـ x (x قيمة مقدرة بالراديان)	cos(0.0) is 1.0
exp(x)	الطريقة الأسية e^x	exp(1.0) is 2.71828 exp(2.0) is 7.38906
floor(x)	تقريب x إلى أكبر عدد صحيح أصغر من x	floor(9.2) is 9.0 floor(-9.8) is -10.0
log(x)	طريقة اللوغاريتم الطبيعي لـ x (للأساس e)	log(Math.E) is 1.0 log(Math.E *Math.E) is 2.0
max(x, y)	طريقة إرجاع القيمة الأكبر بين قيمتين (هذه الطريقة تقبل أيضًا وسائط من النمط float, int, long)	max(2.3, 12.7) is 12.7 max(-2.3, -12.7) is -2.3
min(x, y)	طريقة إرجاع القيمة الأصغر بين قيمتين (هذه الطريقة تقبل أيضًا	min(2.3, 12.7) is 2.3 min(-2.3, -12.7) is -12.7

الطريقة	الوصف	مثال
	وسائط من النمط float, (int, long	
pow(x, y)	طريقة حساب x مرفوعًا إلى القوة y (أي x^y)	pow(2.0, 7.0) is 128.0 pow(9.0, 0.5) is 3.0
sin(x)	طريقة الجيب لـ x (x قيمة مقدرة بالراديان)	sin(0.0) is 0.0
sqrt(x)	الجذر التربيعي للمتغير x	sqrt(900.0) is 30.0 sqrt(9.0) is 3.0
tan(x)	طريقة الظل لـ x (x قيمة مقدرة بالراديان)	tan(0.0) is 0.0

تمتلك هذه المكتبة أيضًا ثوابت عددية هي:

```
Math.E     is 2.7182818284590452354
Math.PI    is 3.14159265358979323846
```

المتغيرات النهائية Final Instance Variables:

تميز معظم لغات البرمجة بين البيانات التي يمكن تغييرها «**المتغيرات**» وبين تلك التي تبقى قيمها ثابتة «**الثوابت**»، لكن هذا التمييز غير موجود في لغة Java لعدم وجود الثوابت، ولا يحد هذا من إمكانيات هذه اللغة التي توفر في الواقع آلية أفضل لا تسمح باستخدام صيغ مشابهة للثوابت فحسب بل تسمح أيضًا بتطبيق هذا المفهوم على عناصر أخرى كالطرق أو الفئات.

لا يمكن تعديل قيمة متغير بعد التصريح عنه بالصفة final، فيلعب بذلك دور الثابت في اللغات الأخرى، وتستخدم هذه المتغيرات غاليًا لتمثيل القيم الثابتة، فإذا

أردت على سبيل المثال أن تستخدم قيمة (π) pi في برنامجك فمن غير المحتمل أن تضطر لتعديلها، لذا يفضل أن تطلق عليها الصفة final:

```
public final double PI = 3.141592653589793;
```

يوفر استخدام هذه المتغيرات ميزتين هامتين، فمن ناحية أولى يزداد عامل الأمان في البرنامج إذ سيرفض المترجم أية محاولة لإسناد قيمة جديدة للمتغير، وتؤدي مثل هذه المحاولة إلى توليد رسالة الخطأ التالية:

```
can't assign a value to a final variable
```

ومن ناحية أخرى، يتحسن مستوى أداء البرنامج إذ يولد المترجم ترميزًا أكثر فعالية عندما يستطيع أن يحدد أن قيمة ما لن يتم تغييرها. وبشكل أكثر تحديدًا يستطيع المترجم عندئذ أن يجري بعض الحسابات في وقت الترجمة بدلاً من تأجيلها إلى وقت التنفيذ، فعندما يصادف المترجم السطر:

```
int x = a + 2;
```

حيث a هو متغير عادي، لن يستطيع إجراء الحساب فورًا ويستبدل بهذا السطر سطر الترميز الموافق، ولن تنفذ العملية إلا في وقت التنفيذ، بالمقابل إذا كان a متغيرًا نهائيًا يعرف المترجم أن قيمته لن تتغير ويستطيع بذلك إجراء الحساب مباشرة وإسناد القيمة المناسبة للمتغير x.

يمكن أن يتضمن البرنامج نوعين من الثوابت: تلك التي تعرف قيمها قبل تنفيذ البرنامج، وتلك التي لا يمكن تحديد قيمها إلا عند تنفيذ البرنامج، ويؤدي استخدام كل نوع إلى تحسين مستوى الأداء بنسبة وآلية مختلفة. ولا يمكن إجراء الحسابات في وقت الترجمة إلا إذا كانت قيم الثوابت معروفة مسبقًا. وتعتمد

الطريقة الأخرى في تحسين مستوى الأداء على تخزين هذه المتغيرات في منطقة من الذاكرة تمكن قراءتها بسرعة لكن بالمقابل يزداد زمن تغيير البيانات المخزنة فيها، وتعتبر هذه الآلية مناسبة للثوابت التي يمكن تحديد قيمها أثناء تنفيذ البرنامج، ويمكن فرض هذه الآلية في Java باستخدام متغيرات نهائية بدون قيم ابتدائية، على أن تسند القيم الابتدائية لهذه المتغيرات لاحقًا ولن يعود ممكنًا بعدئذ تغييرها. ويجري التصريح عن هذه المتغيرات بالصيغة:

```
final double E;
```

ولا يمكن استخدام هذه المتغيرات قبل أن تسند إليها قيمة ابتدائية، ومن الشائع استخدام هذا النمط من المتغيرات للتصريح عن وسطاء الطرق بحيث يقوم المترجم بتحسين ترميز هذه الطرق. وعليه يمكن أن نكتب:

```
int mymethod(final int i, final int y)
{
    body
}
```

فيأخذ المتغيران قيمهما عند استدعاء الطريقة فقط، ولا يعود ممكنًا تعديلهما داخل الطريقة.

إذن فبغض النظر عن نوع المتغير، فإنه يمكن أن يتم تعريفه كمتغير نهائي final.
ولا يمكن تغيير أي متغير تم تعريفه كمتغير نهائي final بمجرد تعيين قيمة له.

تلميح: جرت العادة بين المبرمجين أن يتم تسمية الثوابت بكلمات واضحة ومعبرة، كما جرت العادة على جعل كافة الأحرف في الحالة الكبيرة فمثلاً الاسم PI أفضل من pi للدلالة على الثوابت.

البرنامج التالي يوضح استخدام مكتبة الطرق الرياضية:

```
// Testing the Math class methods.
public class MathTest {
      public static void main( String args[] )
      {
            System.out.println( "Math.abs( 23.7 ) = " +
                  Math.abs( 23.7 ) );

            System.out.println( "Math.abs( 0.0 ) = " +
                  Math.abs( 0.0 ) );

            System.out.println( "Math.abs( -23.7 ) = " +
                  Math.abs( -23.7 ) );

            System.out.println( "Math.ceil( 9.2 ) = " +
                  Math.ceil( 9.2 ) );

            System.out.println( "Math.ceil( -9.8 ) = " +
                  Math.ceil( -9.8 ) );

            System.out.println( "Math.cos( 0.0 ) = " +
                  Math.cos( 0.0 ) );

            System.out.println( "Math.exp( 1.0 ) = " +
                  Math.exp( 1.0 ) );

            System.out.println( "Math.exp( 2.0 ) = " +
                  Math.exp( 2.0 ) );

            System.out.println( "Math.floor( 9.2 ) = " +
                  Math.floor( 9.2 ) );

            System.out.println( "Math.floor( -9.8 ) = " +
                  Math.floor( -9.8 ) );

            System.out.println( "Math.log( Math.E ) = " +
                  Math.log( Math.E ) );
```

```
        System.out.println(
              "Math.log( Math.E * Math.E ) = " +
              Math.log( Math.E * Math.E ) );

        System.out.println( "Math.max( 2.3, 12.7 ) = "+
              Math.max( 2.3, 12.7 ) );

        System.out.println( "Math.max( -2.3, -12.7 )="+
              Math.max( -2.3, -12.7 ) );

        System.out.println( "Math.min( 2.3, 12.7 ) = "+
              Math.min( 2.3, 12.7 ) );

        System.out.println( "Math.min( -2.3, -12.7 )="+
              Math.min( -2.3, -12.7 ) );

        System.out.println( "Math.pow( 2.0, 7.0 ) = " +
              Math.pow( 2.0, 7.0 ) );

        System.out.println( "Math.pow( 9.0, 0.5 ) = " +
              Math.pow( 9.0, 0.5 ) );

        System.out.println( "Math.sin( 0.0 ) = " +
              Math.sin( 0.0 ) );

        System.out.println( "Math.sqrt( 900.0 ) = " +
              Math.sqrt( 900.0 ) );

        System.out.println( " Math.sqrt( 9.0 ) = " +
              Math.sqrt( 9.0 ) );

        System.out.println( " Math.tan( 0.0 ) = " +
              Math.tan( 0.0 ) );

    } // end main
} // end class MathTest
```

فيكون خرج البرنامج كالتالي:

```
Math.abs( 23.7 ) = 23.7
Math.abs( 0.0 ) = 0.0
Math.abs( -23.7 ) = 23.7
Math.ceil( 9.2 ) = 10.0
Math.ceil( -9.8 ) = -9.0
Math.cos( 0.0 ) = 1.0
Math.exp( 1.0 ) = 2.7182818284590455
Math.exp( 2.0 ) = 7.38905609893065
Math.floor( 9.2 ) = 9.0
Math.floor( -9.8 ) = -10.0
Math.log( Math.E ) = 1.0
Math.log( Math.E * Math.E ) = 2.0
Math.max( 2.3, 12.7 ) = 12.7
Math.max( -2.3, -12.7 ) = -2.3
Math.min( 2.3, 12.7 ) = 2.3
Math.min( -2.3, -12.7 ) = -12.7
Math.pow( 2.0, 7.0 ) = 128.0
Math.pow( 9.0, 0.5 ) = 3.0
Math.sin( 0.0 ) = 0.0
Math.sqrt( 900.0 ) = 30.0
Math.sqrt( 9.0 ) = 3.0
Math.tan( 0.0 ) = 0.0
```

التصريح عن الطرق Method Declarations:

يقوم البريمج التالي بحساب مربع الأعداد الصحيحة من ١ إلى ١٠ باستخدام الطريقة square المعرفة من قبل المبرمج:

```
// Creating and using a programmer-defined method.
import java.awt.Container;
import javax.swing.*;

public class SquareIntegers extends JApplet {
//set up GUI and calculate squares of integers from 1to 10
     public void init()
```

```
      {
            // JTextArea to display results
            JTextArea outputArea = new JTextArea();

 // get applet's content pane (GUI component display area)
            Container container = getContentPane();

            // attach outputArea to container
            container.add( outputArea );

            // store result of call to method square
            int result;

            // String containing results
            String output = "";

            // loop 10 times
            for ( int counter=1; counter <= 10; counter++){
                  result = square( counter ); // method call

                  // append result to String output
                  output += "The square of " + counter +
                        " is " + result + "\n";

            } // end for

            // place results in JTextArea
            outputArea.setText( output );

      } // end method init

      // square method declaration
      public int square( int y )
      {
            return y * y; // return square of y
      } // end method square
} // end class SquareIntegers
```

التوضيح: استوردنا المسارين السابقين الأول java.awt.Container وذلك لإنشاء حاوية تحتوي العناصر في البريمج، الثاني javax.swing.* للكائنات الرسومية والفئة JApplet.

قمنا بعد ذلك بتعريف أداة مربع النص متعدد الأسطر:

```
JTextArea outputArea = new JTextArea();
```

ثم أنشأنا الحاوية:

```
Container container = getContentPane();
```

ثم أضفنا أداة مربع النص إلى الحاوية:

```
container.add( outputArea );
```

في حلقة التكرار for استخدمنا السطر:

```
result = square( counter );
```

والذي يعبر عن عملية استدعاء الطريقة square وتمرير المتغير counter كوسيط لهذه الطريقة ليتم استخدام قيمته، وترجع قيمة جديدة من هذه الطريقة لتخزن في المتغير result.

بعد تكوين النص الكامل في المتغير outpot عن طريق الصيغة:

```
output += "The square of " + counter + " is " + result +
"\n";
```

يتم إرسال النص إلى أداة مربع النص متعددة الأسطر outputArea عن طريق التعليمة التالية:

```
outputArea.setText( output );
```

إذ تستخدم الطريقة setText لوضع النص الممرر إليها إلى الأداة النصية.

الشكل العام للتصريح عن طريقة General Format of a Method Declaration:

يمكن تلخيص الشكل العام للتصريح عن الطرق المعرفة من قبل المبرمج:

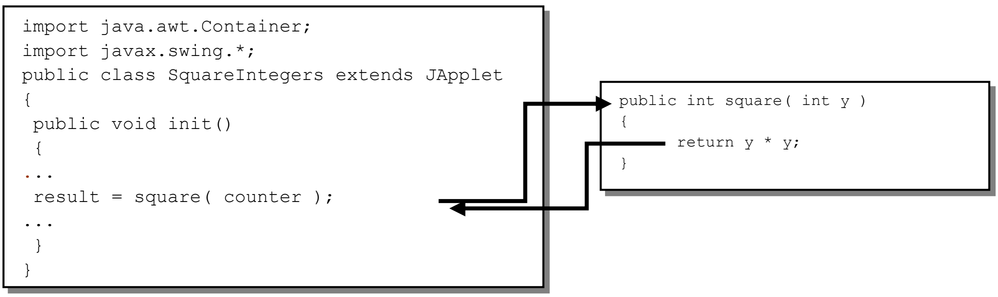
```
return-value-type method-name( parameter1, …,
parameterN )
{
      declarations and statements
}
```

حيث *return-value-type* تعبر عن نوع البيانات التي سترجع من هذه الطريقة ويمكن أن نكتب هنا أي نمط من الأنماط المعروفة سابقاً (أنظر الفصل الثاني). و*method-name* تعبر عن اسم الطريقة وهي تخضع لكافة شروط التسمية في لغة Java (نفس الشروط الموضوعة لأسماء المتغيرات مثلاً). *parameter* تعبر عن الوسائط إذ أن عدد الوسطاء ليس إجباراً وإنما يعتمد على حاجتك والوظيفة المطلوبة من هذه الطريقة. يبدأ جسم الطريقة بالقوس «{» وينتهي بالقوس «}».

في برنامجنا قمنا بكتابة الطريقة square، تتلقى هذه الطريقة نسخة من المتغير counter بواسطة الوسيط y كما هو موضح في طريقة التعريف والاستدعاء السابقة:

```
result = square( counter );
...
public int square( int y )
{
      return y * y;
}
```

بعـدها تقـوم الطريقـة square بحسـاب القيمـة y*y. تجـري إعـادة القيمـة الناتجـة إلى النقطـة الـتي تم عنـدها اسـتدعاء الطريقـة square. يظهـر تعريـف الطريقـة square كمـا لـو أنهـا تنتظـر قيمـة صـحيحة تصـلها مـن خـلال الوسـيط y. تفيـد الكلمـة int الـتي تسـبق اسـم الطريقـة في تحديـد نمـط النتيجـة المعـادة مـن قبـل الطريقـة. في حـين تسـتخدم التعليمـة return مـن أجـل تمريـر النتيجـة إلى الطريقـة الأصـلية الـتي تمـت منهـا عملية الاستدعاء.



استدعاء الطريقة

يوجـد ثلاثـة طـرق لإعـادة الـتحكم إلى النقطـة الـتي تم مـن عنـدها اسـتدعاء إحـدى الطـرق. إذا كانـت الطريقـة لا تعيـد أي قيمـة (الطـرق مـن النـوع void تسـمى في بعـض الأحيـان الإجـراءات)، يمكـن إعـادة الـتحكم بكـل بسـاطة عنـد الوصـول إلى قـوس النهاية «}» الخاص بهذه الطريقة أو عند تنفيذ التعليمة:

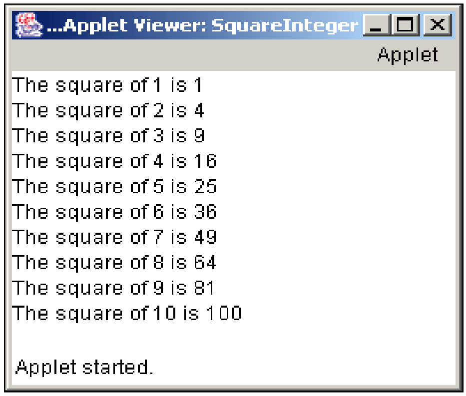
```
return;
```

أما إذا كانت الطريقة تعيد نتيجة فإن التعليمة:

```
return expression;
```

تقوم بإعادة قيمة التعبير expression إلى الطريقة المستدعية.

ونتيجة تنفيذ البريمج السابق تكون كالتالي:



ملحوظة: ❖ يؤدي القيام بالتصريح عن الوسطاء من نفس النمط على الشكل التالي مثلاً: int x, y بدلاً من: int x, int y إلى حدوث خطأ أثناء الترجمة لأنه يجب تحديد نمط كل وسيط من وسطاء الطريقة على حده.

❖ توجد ثلاث حالات لطريقة الاستدعاء:

١. بذكر اسم الطريقة ووسطائها كما في الشكل التالي:

```
x = square( 2 );
```

وهذا يعني استدعاء طريقة موجودة في نفس الفئة class الموجود فيها المستدعي.

٢. استدعاء طريقة خاصة بكائن ما وذلك باستخدام النقطة «.» كالتالي:

```
g.drawString( "Hi" );
```

٣. استدعاء طريقة خاصة بفئة ما (الطرق الساكنة) كالتالي:

```
x = Math.abs( y );
```

نماذج الطرق المعرفة: الطريقة Maximum:

يعــرض الــبريمج التــالي الطريقــة maximum المعرفــة مــن قبــل المــبرمج مــن أجــل تحديد أكبر قيمة من بين ثلاثة قيم صحيحة:

```
// Finding the maximum of three floating-point numbers.
import java.awt.Container;

import javax.swing.*;

public class MaximumTest extends JApplet {
// initialize applet by obtaining user input and
// creating GUI
      public void init()
      {
            // obtain user input
            String s1 = JOptionPane.showInputDialog(
                  "Enter first floating-point value" );

            String s2 = JOptionPane.showInputDialog(
                  "Enter second floating-point value" );

            String s3 = JOptionPane.showInputDialog(
                  "Enter third floating-point value" );

            // convert user input to double values
            double number1 = Double.parseDouble( s1 );
            double number2 = Double.parseDouble( s2 );
            double number3 = Double.parseDouble( s3 );

            // method call
            double max = maximum(number1, number2,number3);
```

```
        // create JTextArea to display results
        JTextArea outputArea = new JTextArea();

        // display numbers and maximum value
        outputArea.setText(   "number1: " + number1 +
             "\nnumber2: " + number2 +
             "\nnumber3: " + number3 +
             "\nmaximum is: " + max );

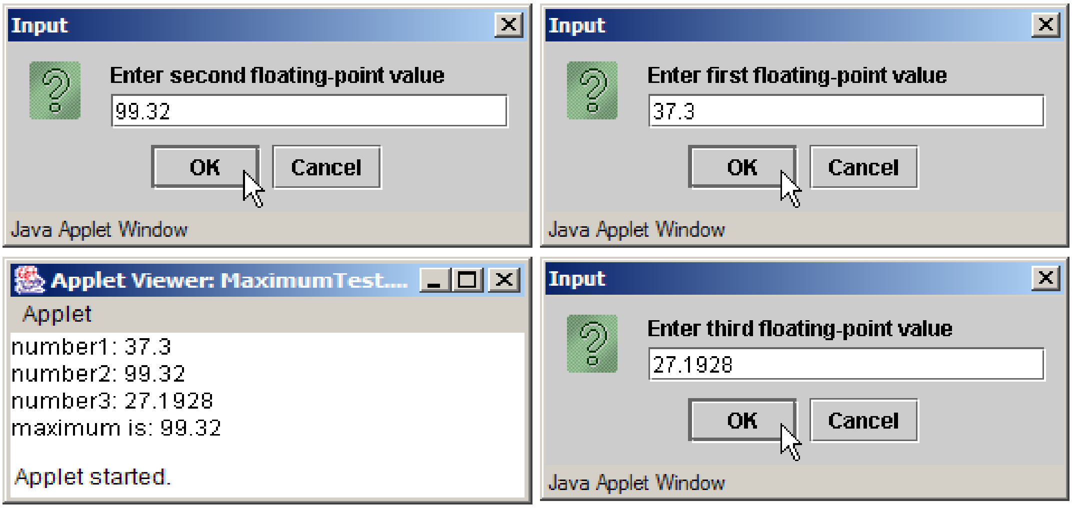
        // get applet's GUI component display area
        Container container = getContentPane();

        // attach outputArea to Container c
        container.add( outputArea );

    } // end method init

    // maximum method uses Math class method max to help
    // determine maximum value
    public double maximum( double x, double y, double z )
    {
        return Math.max( x, Math.max( y, z ) );
    } // end method maximum
} // end class Maximum
```

وناتج تنفيذ هذا البريمج كالتالي:



قسر الوسطاء Argument Promotion or Coercion:

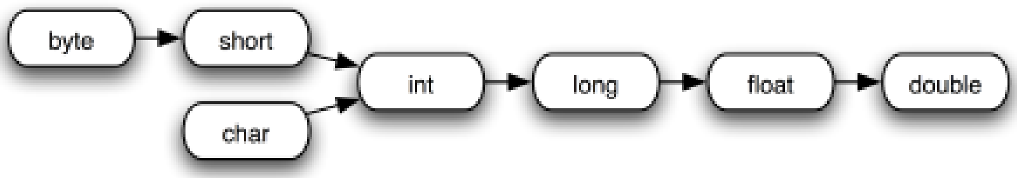
يعبر هذا المصطلح عن إجبار الوسطاء على أن تصبح من نمط معين مناسب. على سبيل المثال، يمكن استدعاء الطريقة الرياضية sqrt التي تحسب الجذر التربيعي مع الأعداد الصحيحة رغم أنها تقبل الوسطاء من النوع double في تعريفها، أي أن التعليمة:

```
System.out.println( Math.sqrt( 4 ) );
```

تعمل بشكل صحيح لتطبع القيمة ٢ حيث يقوم نموذج الطريقة بإجبار المترجم على تحويل القيمة الصحيحة ٤ إلى القيمة الحقيقية ٤,٠ قبل أن يمررها إلى جسم الطريقة نفسها..والجدول التالي يوضح هيكلية قسر أنماط البيانات الجاهزة في لغة Java:

نمط البيانات	إمكانية القسر إلى
double	none
float	double
long	double, float

نمط البيانات	إمكانية القسر إلى
int	double, float, long
char	double, float, long, int
short	double, float, long, int
byte	double, float, long, int, short
boolean	none



توليد الأعداد العشوائية Random-Number Generation:

تستخدم الطريقة ()Math.random لتوليد أرقام عشوائية من النمط double محصورة بين الرقمين ٠ و١. ولجعل المدى يبعد إلى قيمة ما ولتكن m نقوم بحساب القيمة التالية:

```
c = Math.random() * m;
```

تعبر هذه التعليمة عن عملية توليد أرقام عشوائية محصورة بين الرقمين ٠ وm. ولتحريك المدى نقوم مثلاً بكتابة التعليمة التالية:

```
c = Math.random() + m;
```

لتوليد أرقام عشوائية محصورة بين الرقمين m وm+1. ولبناء قاعدة أساسية يمكن استخدام التعليمة التالية:

```
c = ( Math.random() * ( m - n ) ) + n;
```

تولد هذه التعليمة أرقامًا عشوائيةً محصورة بين الرقمين n وm ، أي أن أقل رقم عشوائي يمكن توقعه هو الرقم n وأكبر رقم عشوائي يمكن ظهوره هو m.

يتغير الوضع عندما نرغب في توليد أرقام عشوائية صحيحة ، عندها ينقص ١ عند إجراء عملية التوليد ويجب إضافته لكي تكون العملية صحيحة ، أي تصبح التعليمة السابقة كالتالي:

```
c = (int)( Math.random() * ( m - n + 1 ) ) + n;
```

استخدمنا التعليمة (int) أما العبارة لتحويل القيمة إلى قيمة صحيحة.

البرنامج التالي يوضح عملية استخدام الطريقة random في عملية توليد أرقام عشوائية:

```
import javax.swing.JOptionPane;

public class RandomIntegers {
      public static void main( String args[] )
      {
            int value, n = 7, m = 9;
            String output = "";

            // loop 20 times
            for (int counter = 1; counter <= 20;counter++){
                  // pick random integer between n and m
                  value = (int)( Math.random() *
                        (m - n + 1) ) + n ;

                  // append value to output
                  output += value + " ";

                  // if counter divisible by 5,
                  // append newline to String output
```

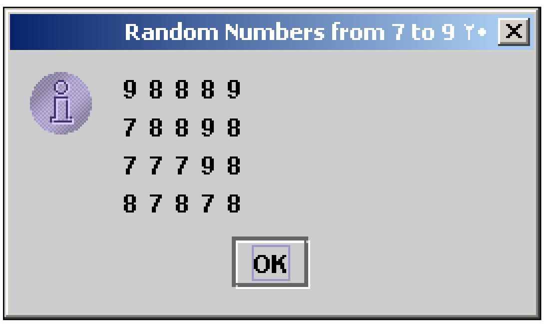
```
                if ( counter % 5 == 0 )
                    output += "\n";

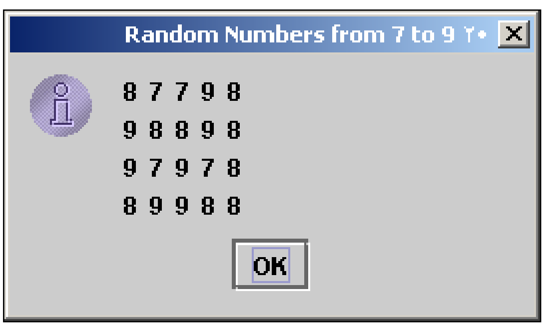
            } // end for

            JOptionPane.showMessageDialog( null, output,
                "20 Random Numbers from " + n + " to " +
                m, JOptionPane.INFORMATION_MESSAGE );

            System.exit( 0 ); // terminate application
        } // end main
} // end class RandomIntegers
```

فيكون خرج البرنامج كالتالي في حالة تشغيله مرتين:





البرنامج التالي يقوم باختبار احتمالية مرات تكرار ظهور إحدى أوجه حجر النرد الستة:

```
// Roll a six-sided die 6000 times.
import javax.swing.*;

public class RollDie {
      public static void main( String args[] )
      {
            int frequency1 = 0, frequency2 = 0,
                  frequency3 = 0,  frequency4 = 0,
                  frequency5 = 0, frequency6 = 0, face;

            // summarize results
            for ( int roll = 1; roll <= 6000; roll++ ) {
                  face = 1 + ( int ) ( Math.random() * 6 );

                  // determine roll value and increment
                  // appropriate counter
                  switch ( face ) {
                        case 1:
                              ++frequency1;
                              break;

                        case 2:
                              ++frequency2;
                              break;

                        case 3:
                              ++frequency3;
                              break;

                        case 4:
                              ++frequency4;
                              break;

                        case 5:
                              ++frequency5;
                              break;

                        case 6:
```

```
                              ++frequency6;
                              break;
                    } // end switch
               } // end for

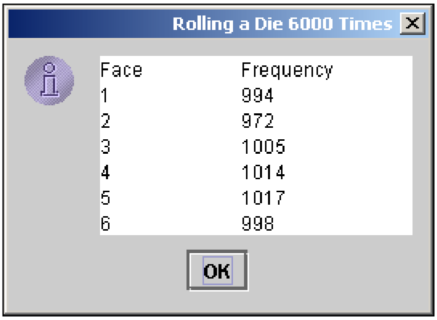
               JTextArea outputArea = new JTextArea();

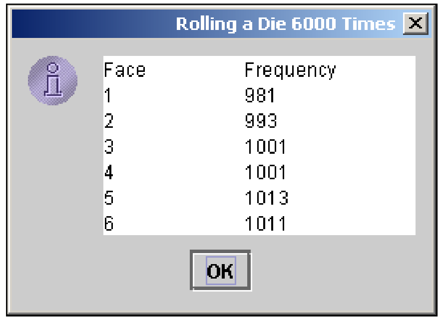
               outputArea.setText( "Face\tFrequency" + "\n1\t"
                    + frequency1 +  "\n2\t" + frequency2 +
                    "\n3\t" + frequency3 + "\n4\t" +
                    frequency4 + "\n5\t" + frequency5 +
                    "\n6\t" + frequency6 );

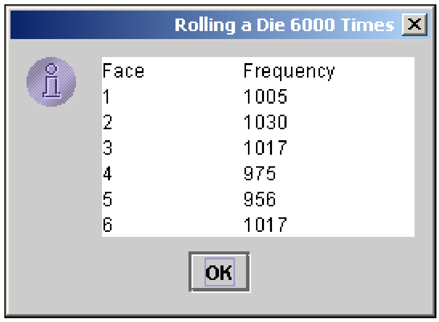
               JOptionPane.showMessageDialog( null,
                    outputArea, "Rolling a Die 6000 Times",
                    JOptionPane.INFORMATION_MESSAGE );

               System.exit( 0 ); // terminate application
          } // end main
} // end class RollDie
```

وعند تنفيذه لأكثر من مرة يكون الخرج كالتالي:

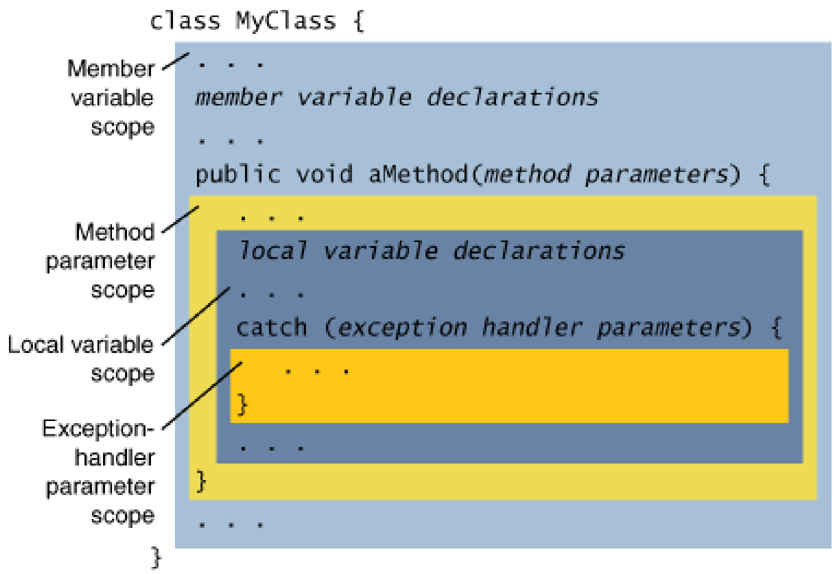

Rolling a Die 6000 Times
Face Frequency
1 994
2 972
3 1005
4 1014
5 1017
6 998
OK


Rolling a Die 6000 Times
Face Frequency
1 981
2 993
3 1001
4 1001
5 1013
6 1011
OK


Rolling a Die 6000 Times
Face Frequency
1 1005
2 1030
3 1017
4 975
5 956
6 1017
OK

قواعد مجال الرؤية Scope Rules:

يتحدد مجال الرؤية المتعلق بمُعرِف ما بجزء البرنامج الذي نستطيع ضمنه التعامل مع هذا المُعرف. على سبيل المثال عندما نصرح عن متغير محلي ضمن كتلة فيمكن التعامل معه فقط ضمن هذه الكتلة.



يوجد أنواع من مجالات الرؤية لمُعرف ما ومنها:

- مجال الرؤية على مستوى الطريقة method scope
- مجال الرؤية على مستوى الملف file scope
- مجال الرؤية على مستوى الكتلة block scope

والبريمج التالي لتوضيح الفكرة:

```
// A scoping example.
import java.awt.Container;
import javax.swing.*;
```

```
public class Scoping extends JApplet {
      JTextArea outputArea;

      // field that is accessible to all
      // methods of this class
      int x = 1;

      // create applet's GUI
      public void init()
      {
            outputArea = new JTextArea();
            Container container = getContentPane();
            container.add( outputArea );

      } // end method init

      // method start called after init completes;
      // start calls
      // methods useLocal and useField
      public void start()
      {
            int x = 5; //local variable in method start
                       // that shadows field x

            outputArea.append( "local x in start is " +
                  x );

            useLocal(); // useLocal has local x
            useField(); // useInstance uses Scoping's
                        // field x
            useLocal(); // useLocal reinitializes local x
            useField(); // Scoping's field x retains
                        // its value

            outputArea.append( "\n\nlocal x in start is " +
                  x );
```

```
    } // end method start

    // useLocal creates and initializes local
    // variable x during each call
    public void useLocal()
    {
          int x = 25; // initialized each time
                      // useLocal is called

          outputArea.append("\n\nlocal x in useLocal is "
                + x + " after entering useLocal" );

          ++x;

          outputArea.append( "\nlocal x in useLocal is "
                + x + " before exiting useLocal" );

    } // end method useLocal

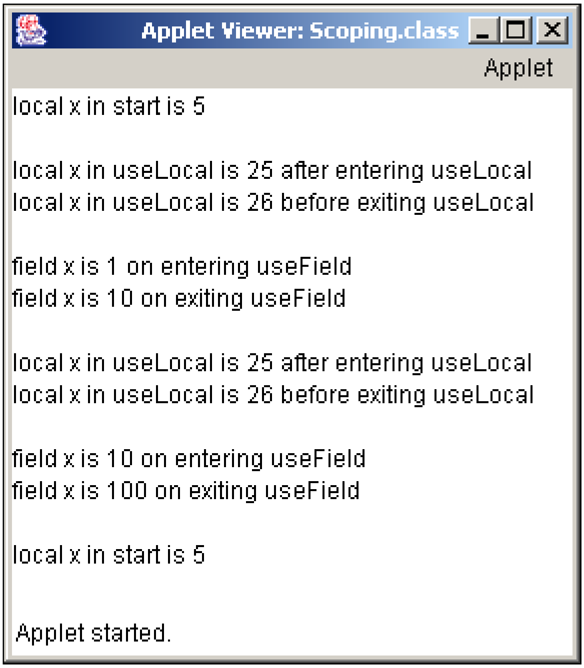
    // useField modifies Scoping's field x during
    // each call
    public void useField()
    {
          outputArea.append( "\n\nfield x is " + x +
                " on entering useField" );

          x *= 10;

          outputArea.append( "\nfield x is " + x +
                " on exiting useField" );

    } // end method useInstance
} // end class Scoping
```

وعند تنفيذه يكون خرجه كالتالي:



التحميل الزائد للطرق Method Overloading:

تسمح لغة Java بتعريف عدة طرق لها نفس الاسم بشرط أن تتعامل مع وسطاء لها أنماط مختلفة. تسمى هذه الإمكانية «**التحميل الزائد للطرق** Method Overloading». عند استدعاء طريقة مُزادة overloaded فإن المترجم يقوم باختيار الطريقة المناسبة بعد التحقق من عدد ونمط وترتيب الوسطاء الخاصة بعملية الاستدعاء نفسها. يتم استخدام إمكانية التحميل الزائد للطرق لبناء عدة طرق لها نفس الاسم وتقوم بنفس المهمة أيضًا ولكن مع أنماط مختلفة للبيانات.

البريمج التالي يستخدم التحميل الزائد للطرق من أجل حساب مربع عدد من النمط int ومن النمط double:

```
// Using overloaded methods
import java.awt.Container;
```

```
import javax.swing.*;

public class MethodOverload extends JApplet {
      // create GUI and call each square method
      public void init()
      {
            JTextArea outputArea = new JTextArea();
            Container container = getContentPane();
            container.add( outputArea );

            outputArea.setText("The square of integer 7 is"
                  + square( 7 ) +
                  "\nThe square of double 7.5 is " +
                  square( 7.5 ) );

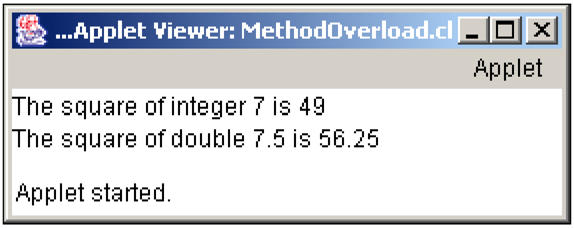
      } // end method init

      // square method with int argument
      public int square( int intValue )
      {
            System.out.println(
                  "Called square with int argument: " +
                  intValue );

            return intValue * intValue;
      } // end method square with int argument

      // square method with double argument
      public double square( double doubleValue )
      {
            System.out.println(
                  "Called square with double argument: " +
                  doubleValue );

            return doubleValue * doubleValue;
      } // end method square with double argument
} // end class MethodOverload
```




```
Called square with int argument: 7
Called square with double argument: 7.5
```

البريمج التالي يوضح أنه عند إجراء التحميل الزائد للطرق لابد أن تختلف الوسائط بين الطرق المزودة:

```
// Overloaded methods with identical signatures.
import java.awt.Container;
import javax.swing.*;

public class MethodOverload extends JApplet {
      // create GUI and call each square method
      public void init()
      {
            JTextArea outputArea = new JTextArea();
            Container container = getContentPane();
            container.add( outputArea );

            outputArea.setText( "The square of 7 is " +
                  square( 7 ) );
      } // end method init

      // declaration of method square with int argument
      public int square( int x )
      {
            return x * x;
      }

      // second declaration of method square
      // with int argument causes syntax error
      public double square( int y )
```

```
        {
                return y * y;
        }
} // end class MethodOverload
```

وعند تنفيذ البرنامج السابق يكون الخرج كالتالي:

```
MethodOverload.java:24: square(int) is already defined in
MethodOverload
 public double square( int y )

1 error
```

الاستدعاء الذاتي (العودية) Recursion:

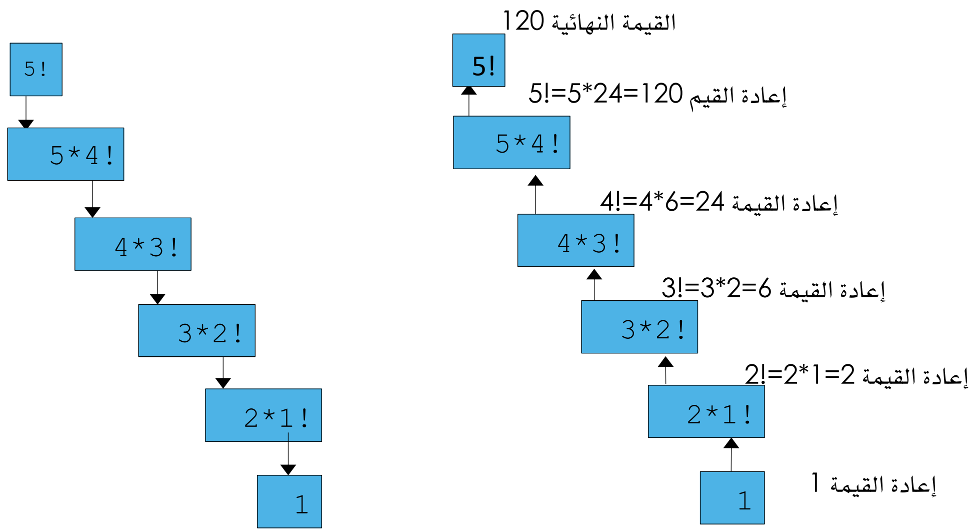
تتألف البرامج التي قمنا بعرضها فيما سبق من مجموعة من الطرق التي تقوم باستدعاء بعضها البعض بأسلوب منهجي وبطريقة هرمية. لكن من المفيد في بعض المسائل، أن تقوم بعض الطرق باستدعاء نفسها. نسمي الطرق السابقة بطرق الاستدعاء الذاتي (الطرق العودية) recursive methods لأنها كما قلنا تستدعي نفسها بشكل مباشر أو غير مباشر من خلال طريقة أخرى.



تتمتع أساليب الحل لمسائل تستدعي استخدام عملية الاستدعاء الذاتي بعدد من العناصر المشتركة حيث يتم استدعاء طريقة عودية تعرف فقط كيفية حل المسألة في أبسط حالاتها. نسمي هذه الحالة أو الحالات البسيطة «بالحالة أو الحالات الابتدائية base cases». أي عند استدعاء الطريقة مع حالة ابتدائية فإنه يعيد

ببساطة الحل المتعلق بهذه الحالة. أما عند استدعائها مع حالة أكثر تعقيداً فإنها تقوم بتقسيم هذه الحالة إلى جزأين بحيث تعرف ماذا تفعل بالجزء الأول ولكنها لا تعلم ماذا يجب عمله مع الجزء الثاني. لكن من أجل إيقاف العودية، يجب على الجزء الثاني أن يشابه المسألة الأصلية لكنه يختلف عنها في كونه يتناول حالة أسهل وأبسط من حالة المسألة الأصلية. وعلى اعتبار أن حالة الجزء الثاني تتناول مسألة مشابهة للمسألة الأصلية لذلك يمكن للطريقة أن تعود لتستدعي نفسها لحل مسألة أصغر من المسألة السابقة.. وهكذا. نسمي هذه العودية في الاستدعاء recursive call بخطوة العودية recursion step. تتضمن خطوة العودية على العبارة return لأنه يجب على نتائج هذه الخطوة أن تنضم إلى نتائج الجزء الأول الذي تعرف الطريقة العودية التعامل معه من أجل تكوين حل للمسألة المتناولة وإعادة النتيجة إلى الطريقة المستدعية. ولتوضيح ذلك سندرس مثال على الاستدعاء الذاتي وهو حساب المضروب factorial إذا كان لدينا n عدد صحيح موجب، فإن القيمة n! (تقرأ مضروب العدد n) تساوي قيمة الجداء:

```
n!=n * (n-1) * (n-2) * …* 1
1!=1
0!=1
n! = n * (n-1)!
```



أ) معالجة الاستدعاء العودي

ب) القيم المعادة من الاستدعاءات العودية

والبرنامج التالي يظهر استخدام عملية الاستدعاء الذاتي في حل هذا النوع من المسائل:

```
// Recursive factorial method.
import java.awt.*;
import javax.swing.*;

public class FactorialTest extends JApplet {
      JTextArea outputArea;

      // create GUI and calculate factorials of 0-10
      public void init()
      {
            outputArea = new JTextArea();

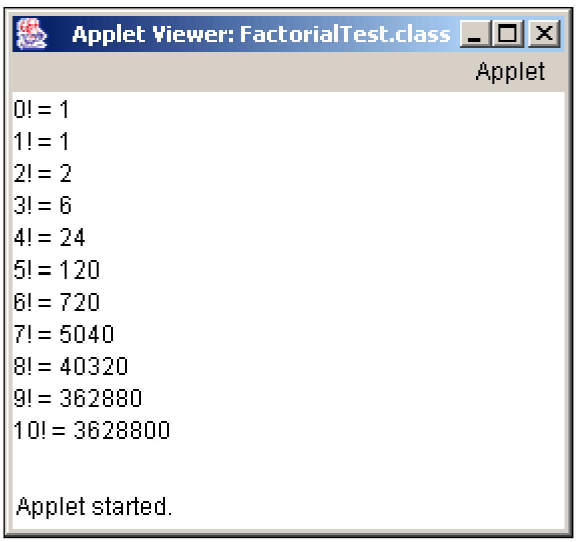
            Container container = getContentPane();
            container.add( outputArea );

            // calculate the factorials of 0 through 10
            for(long counter = 0;counter <= 10; counter++ )
```

```
                  outputArea.append( counter + "! = " +
                        factorial( counter ) + "\n" );
      } // end method init

      // recursive declaration of method factorial
      public long factorial( long number )
      {
            // base case
            if ( number <= 1 )
                  return 1;
            // recursive step
            else
                  return number * factorial( number - 1 );

      } // end method factorial
} // end class FactorialTest
```



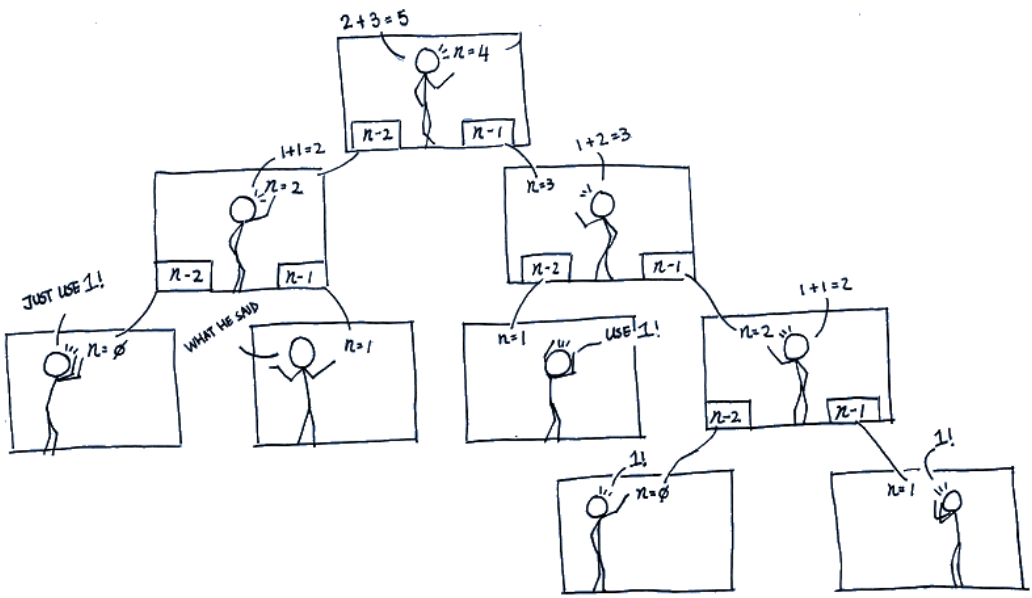
مثال آخر عن الاستدعاء الذاتي: سلاسل Fibonacci:

تبدأ سلاسل Fibonacci التالية:

```
0, 1, 1, 2, 3, 5, 8, 13, 21, …
```

مـع القيمـتين ٠ و١ وتتمتـع بالخاصـية التاليـة: يسـاوي كـل عـدد مـن أعـدادها مجمـوع العددين السابقين له.

```
fibonacci( n ) = fibonacci( n - 1 ) + fibonacci( n - 2 )
```



البريمج التالي يوضح استخدام الطريقة fibonacci:

```
// Recursive fibonacci method.
import java.awt.*;
import javax.swing.*;

public class FibonacciTest extends JApplet {
      JTextArea outputArea;

```

```
      // create GUI and calculate fibonaccis of 0-10
      public void init()
      {
            outputArea = new JTextArea();

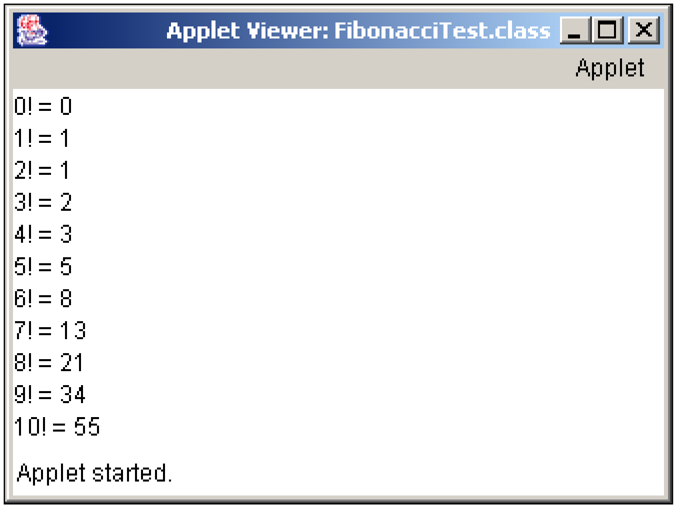
            Container container = getContentPane();
            container.add( outputArea );

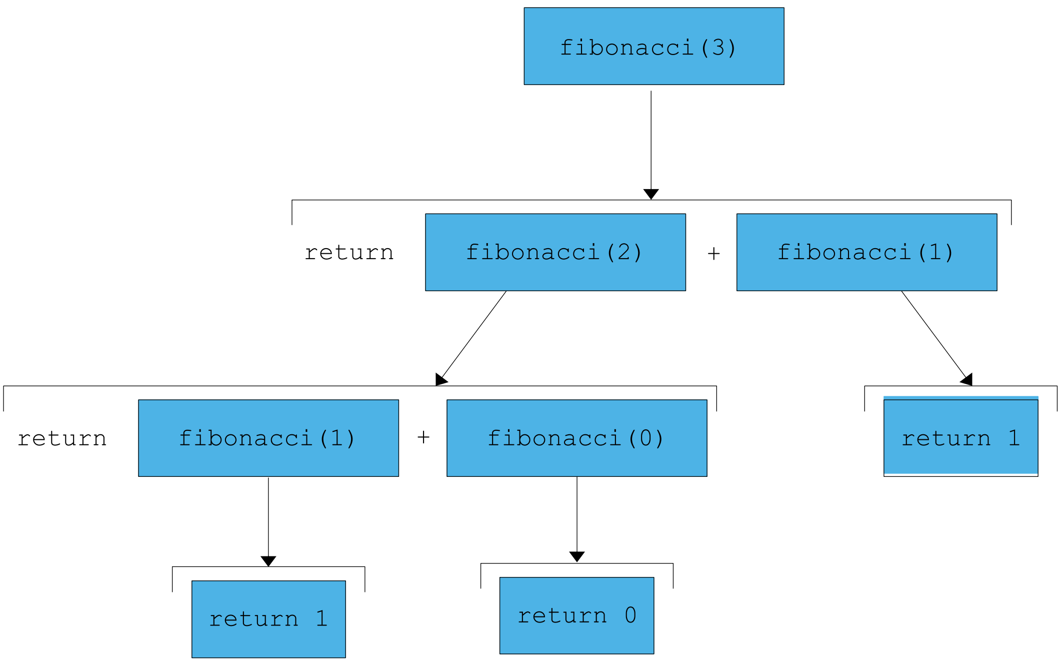
            // calculate the fibonaccis of 0 through 10
            for(long counter = 0;counter <= 10; counter++ )
                  outputArea.append( counter + "! = " +
                        fibonacci( counter ) + "\n" );

      } // end method init

      // recursive declaration of method fibonacci
      public long fibonacci( long n )
      {
            // base case
            if ( n == 0 || n == 1 )
                  return n;
            // recursive step
            else
                  return fibonacci( n - 1 ) +
                        fibonacci( n - 2 );

      } // end method fibonacci
} // end class FibonacciTest
```


Applet Viewer: FibonacciTest.class
Applet
0! = 0
1! = 1
2! = 1
3! = 2
4! = 3
5! = 5
6! = 8
7! = 13
8! = 21
9! = 34
10! = 55
Applet started.


fibonacci(3)
return
fibonacci(2)
+
fibonacci(1)
return
fibonacci(1)
+
fibonacci(0)
return 1
return 1
return 0

تمارين الفصل الخامس:

١ . ما الفائدة من استخدام الطرق في البرنامج؟

٢ . ماذا نقصد بمصطلح مجال الرؤية وأذكر بعضًا منها؟

٣ . اذكر أمثلة رياضية لاستخدام عملية التحميل الزائد؟

٤. املأ الفراغات التالية:

- المكونات الأساسية لبرامج لغة Java هي.................. و...................
- يتم تنفيذ تعليمات الطريقة بعد القيام بـ...................
- نسمي المتغير المعروف فقط ضمن جسم الطريقة حيث تم التصريح عنه بـ...................
- تستخدم الكلمة الأساسية.................. ضمن الطريقة المستدعاة من أجل إعادة قيمة منها إلى الطريقة المستدعية.
- يساعد المحدد.................. على التصريح عن متغيرات قابلة للقراءة فقط أو الكتابة فيها لمرة واحدة فقط.

٥ . اكتب الطريقة myPow(base, exponenr) التي تعيد قيمة العبارة: $base^{exponenr}$

فعلى سبيل المثال فإن الاستدعاء myPow(3,4) يعني حساب القيمة ٣×٣×٣×٣.

يجب كتابة الطريقة السابقة، بفرض أن exponenr هو عبارة عن عدد صحيح موجب. استخدم إحدى بنى التكرار لتنفيذ ذلك. لا تستخدم أي طريقة من الطرق الجاهزة.

٦ . اعد كتابة الطريقة myPow بحيث تدعم الاستدعاء الذاتي. (مفتاح الحل هو: $x^y = x * x^{y-1}$).

٧ . اكتب الطريقة isPrime التي تأخذ عددًا صحيحًا وتعيد قيمة منطقية (true; false) تفيد بأن هذا العدد هو عدد أولي أو لا. حيث أن الأعداد الأولية هي تلك الأعداد التي لا تقبل القسمة إلا على نفسها والواحد.

٨ . اكتب طريقة تأخذ عددًا صحيحًا وتعيد أرقام هذا العدد بصورة معكوسة. على سبيل المثال، إذا أعطينا الطريقة الرقم ٧٣٥٤ تعيد ٤٥٣٧.

٩ . اكتب الطريقة mylog(base, num) التي تحسب اللوغاريتم للأساس المحدد كالتالي: $Log_{base}(num)$.

١٠ . اكتب طريقة تستقبل من المستخدم عددين صحيحين وتعيد رقمًا عشوائيًا من النمط double محصور بين هذين العددين.

١١ . طور الطريقة mypow بحيث تدعم الأسس السالبة، والعشرية أي يمكنها حساب: mypow(2.0,-1) وmypow(4.0,0.5).

١٢ . اكتب طريقة لحساب قيمة الثابت PI، حيث تستقبل هذه الطريقة القيمة n لتنفذ المعادلة الحسابية التالية:

$$\pi = \sum_{i=0}^{n} (-1)^i \frac{4}{2i+1} = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + ... + \frac{(-1)^n 4}{2n+1}$$

قم بإنشاء المتغير النهائي myPI واسند إليه قيمة هذه الطريقة.

الملاحق

امتحان نهائي نموذج (١)

مدة الامتحان: ساعتين

أجب عن جميع الأسئلة التالية:

السؤال الأول [15 درجة]:

الفقرة الأولى: عرف ما يلي (بشكل مختصر) [5 درجات]:

آلة جافا الافتراضيـــــية Java Virtual Machine (JVM) - أســـــلوب البرمجة المهيكلة Structured Programming - لغة النمذجة الموحدة UML - التحميل الزائد Overloading.

الفقرة الثانية: أكمل العبارات التالية [5 درجات]:

١. يســـتخدم الأمر .... لترجمة ملفات لغة Java. ويتم إنشـــاء ملف يمتلك الامتداد ....

٢. يتم تنفيذ الطريقة .... تلقائيًا عندما يدخُل البريمج في حالة اللافعالية.

٣. يمكن كتابة جميع البرامج باستخدام ثلاثة أنواع من بنى التحكم: .... و .... و ....

الفقرة الثالثة: وضح الفرق بين كلاً من: (مع التوضيح المطلوب) [5 درجات]:

١. تطبيقات جافا و بريمجيات جافا (وضح بمثال بسيط جدًا).

٢. بنية الاختيار وبنية التكرار (وضـــح باســـتخدام تمثيل UML لحالة واحدة لكل بنية).

السؤال الثاني [21 درجة]:

الفقرة الأولى: اكتب تعليمة أو مجموعة تعليمات بلغة Java للقيام بالمهام التالية [7 درجات]:

١. التصريح عن المتغير c من النمط الحرفي وتهيئته بالقيمة A
٢. تمثيل المعادلة الحسابية التالية: $y = \sin(\ln(x^7))$
٣. رسم مربع يبدأ من النقطة (0, 0) وبطول ضلع مقداره 100
٤. التصريح عن الثابت E من النمط double وبالقيمة 2.71828182

الفقرة الثانية: ما هو خرج الأجزاء البرمجية التالية [7 درجات]:

.١

```
for( int i = 1; i <= 10; i += 2 ){
      if( i % 3 != 0)
            continue;
      else if( i % 10 == 0)
            break;
      else
            System.out.println(i);
}
```

.٢

```
int i = 5, j = 1;
System.out.println( i += 2 );
System.out.println( j -= i++ );
System.out.println( i = 2 + --j );
System.out.println( --i );
System.out.println( i );
```

الفقرة الثالثة: ارسم مخطط UML الخاص بالتعليمات التالية [7 درجات]:

١.

```
if( x == 5 )
      for( i = 0; i<=10 ; i++)
            System.out.println( i += 2 );
```

٢.

```
do{
      switch(i){
            case 1:
            case 2:
                  System.out.println( "A" );
                  break;
            default:
                  System.out.println( "B" );
      }
}while(i <= 10);
```

السؤال الثالث [24 درجة]:

الفقرة الأولى: اكتب البرامج التالية [12 درجة]:

١. برنامجاً يستقبل من المستخدم رقمين وإشارة عملية حسابية (مثل `+, -, *, /`) ثم يقوم بحساب ناتج تنفيذ هذه العملية على الرقمين (استخدم بنية الاختيار `switch`).

٢. برنامجاً لحساب ناتج المعادلة الرياضية التالية:

$$y = \sum_{i=0}^{10} x^i + \frac{4}{i!}$$

الفقرة الثانية: اكتب الطرق التالية [12 درجة]:

١ . الطريقة isPrime(intvalue) التي تأخذ عددًا صـــحيحًا وتعيد قيمة منطقية. صـــواب true في حالة كان هذا العدد الوســـيط أوليًا أو خطأ false في حالة لم يكن العدد أوليًا. (العدد الأولي هو العدد الذي لا يقبل القســـمة إلا على نفســـه وعلى الواحد بدون باقي).

٢ . الطريقة logn(base,number) تقوم هذه الطريقة بحســـــــاب قيمة اللوغاريتم للأســـاس base بالشـــكل التالي: $Log_{base}(x)$. (علمًا بأن $Log_a(x) = \frac{Ln(x)}{Ln(a)}$ ، حيث Ln هو اللوغاريتم الطبيعي أي للأساس e).

امتحان نهائي نموذج (٢)

مدة الامتحان: ساعتين

أجب عن جميع الأسئلة التالية:

السؤال الأول [15 درجة]:

الفقرة الأولى: عرف ما يلي (بشكل مختصر) [5 درجات]:

الشفرة البايتية ByteCode - الخوارزمية Algorithms - قسر الوسطاء Argument Coercion - الاستدعاء الذاتي Recursion.

الفقرة الثانية: أكمل العبارات التالية [5 درجات]:

١. تستخدم الطريقة (الدالة) .... لطباعة وسيطها كنص على مجرى الخرج القياسي.

٢. لتنفيذ البريمج ضمن صفحات الويب نستخدم الوسم (العلامة) .... في لغة البرمجة HTML.

٣. تستخدم الكلمة الأساسية .... ضمن الطريقة المستدعاة من أجل إعادة قيمة منها إلى الطريقة المستدعية.

الفقرة الثالثة: وضح الفرق بين كلاً من: (مع التوضيح المطلوب) [5 درجات]:

١. العبارة For و العبارة Do/While (وضح ذلك بمثال لكل منهما).

٢. الطريقة التي تمتلك نمط البيانات void والتي تمتلك نوع بيانات معين (مثل int) (وضح ذلك بمثال لكل منهما).

السؤال الثاني [21 درجة]:

الفقرة الأولى: اكتب تعليمة أو مجموعة تعليمات بلغة Java للقيام بالمهام التالية [7 درجات]:

١. التصريح عن المتغير x من النمط "عدد صحيح طويل" وتهيئته بالقيمة 123

٢. تمثيل المعادلة الحسابية التالية: $y = \sqrt[3]{e^x}$

٣. رسم خط مستقيم أفقي يبدأ من النقطة (10, 10) وبطول مقداره 100

٤. التصريح عن الثابت PI من النمط double وبالقيمة 3.14159265

الفقرة الثانية: ما هو خرج الأجزاء البرمجية التالية [7 درجات]:

١.

```
for( int i = 1; i <= 10; i += 0 )
{
      if( ++i % 2 == 0)
            continue;
      else
            System.out.println(i++);
}
```

٢.

```
int i = 1;
do{
      switch(i){
            case 1:
            case 2:
                  System.out.println( i-- );
                  break;
            case 3:
                  System.out.println( ++i );
                  break;
      }
}while( i % 2 != 0 );
```

الفقرة الثالثة: اكتب مخطط UML الخاص بالتعليمات التالية [7 درجات]:

.١

```
for( i = 0; i<=10 ; i++)
      if(i % 2 == 0)
            System.out.println( i += 2 );
```

.٢

```
while(i <= 10){
      if(i == 1)
            System.out.println( i += 10 );
      else if (i == 2)
            System.out.println( i += 20 );
      else
            System.out.println( i );
}
```

السؤال الثالث [24 درجة]:

الفقرة الأولى: اكتب البرامج التالية [12 درجة]:

١. برنامجاً يستقبل من المستخدم رقمين x وy ورقم للترتيب (إما ١ أو ٢)، ثم يقوم بحساب ناتج تنفيذ العملية بحسب رقم الترتيب الرقم ١ لحساب x^y والرقم ٢ لحساب y^x. (استخدم بنية الاختيار if/else).

٢. برنامجاً لحساب ناتج المعادلة الرياضية التالية:

$$y = \prod_{i=1}^{10} x^i$$

الفقرة الثانية : اكتب الطرق التالية [12 درجة] :

١. الطريقة **Pow(base, exponenr)** التي تعيد قيمة العبارة: $\mathbf{base}^{\mathbf{exponenr}}$. حيث أن **exponenr** هو عدد صحيح موجب.

٢. الطريقة **sgn** التي تأخذ عدد كوسيط لها ، وتعيد القيمة **0** في حالة كان العدد **0** والقيمة **1** في حالة كان العدد موجبًا والعدد **1-** في حالة كان العدد سالبًا.

مع تمنياتنا لكم بالتوفيق والنجاح

م. أحمد جبر

أهم المصادر والمراجع

أولاً: أهم المصادر والمراجع العربية:

- بطيخ، جمال: **برمجة قواعد البيانات باستخدام ORACLE و JAVA**. حلب: شعاع للنشر والتوزيع، ٢٠٠٥.
- حجار، فادي: **لغة البرمجة JAVA 2**. حلب: شعاع للنشر والتوزيع، ٢٠٠١.
- حجار، فادي: **UML لغة النمذجة الموحدة لمبرمجي جافا**. حلب: شعاع للنشر والتوزيع، ٢٠٠٥.
- حسن، حسن شاليش والنقري، مهيب (ترجمة وإعداد): **البرمجة بلغة جافا باستخدام JBUILDER 4 BORLAND جـ١ : مدخل إلى JBUILDER 4**. دمشق: دار الرضا للنشر، ٢٠٠١.
- حسن، حسن شاليش والنقري، مهيب (ترجمة وإعداد): **البرمجة بلغة جافا باستخدام JBUILDER 4 BORLAND جـ٢ : دليل المطورين**. دمشق: دار الرضا للنشر، ٢٠٠١.
- الحنبظلي، عبد الحميد: **الأسئلة الأكثر طرحًا حول JAVA**. حلب: شعاع للنشر والتوزيع، ٢٠٠٦.
- خرسة، جلال (ترجمة وإعداد): **تعلم JAVA باستخدام بيئة التطوير JBUILDER**. حلب: شعاع للنشر والتوزيع، ٢٠٠٦.
- خرسة، جلال: **لنبدأ مع JAVA : خطواتك الأولى لإتقان JAVA بسهولة ويسر**. حلب: شعاع للنشر والتوزيع، ٢٠٠٦.
- دوناهو، جون: **برمجة قواعد البيانات باستخدام جافا بايبل JAVA DATABASE PROGRAMMING BIBLE**. ترجمة: قسم الترجمة بدار الفاروق وخالد العامري القاهرة: دار الفاروق للنشر والتوزيع، ٢٠٠٣.

- الدوه جي، صلاح (ترجمة وإعداد): **كيف تبرمج بلغة C++**، الطبعة الأولى. حلب: شعاع للنشر والتوزيع، ١٩٩٩.
- الدوه جي، صلاح (ترجمة وإعداد): **كيف تبرمج بلغة C++**، الطبعة الثانية. حلب: شعاع للنشر والتوزيع، ٢٠٠٤.
- شمسي، همام: **تطبيقات قواعد البيانات باستخدام Java**. حلب: شعاع للنشر والتوزيع، ٢٠٠٤.
- الشمعة، خالد عبد السائر والعبد الله، أسامة: **JSP كتاب المبرمج: عرض شامل لبريمجات Servlet 2.4 وبروتوكول HTTP و JSP 2.0 وتغطية لنماذج HTML وتقنية JDBC**. حلب: شعاع للنشر والتوزيع، ٢٠٠٥.
- عزب، عزب محمد وماجد، مصطفى وحسين، قدري طلعت: **الطريق إلى احتراف لغة Java**. القاهرة: دار الكتب العلمية للنشر والتوزيع، ٢٠٠٢.
- عزب، عزب محمد وماجد، مصطفى وحسين، قدري طلعت: **تعلم واحترف Java بسهولة**. القاهرة: دار الكتب العلمية للنشر والتوزيع، ٢٠٠٣.
- القضماني، رامي (ترجمة وإعداد): **تعلم C#**. حلب: شعاع للنشر والتوزيع، ٢٠٠٣.
- لينيكر، ريك: **علم نفسك Visual J++ 6 في ٢١ يومًا**. بيروت: الدار العربية للعلوم، ٢٠٠٠.
- محفوظ، محمد مناف (ترجمة وإعداد): **برمجة الجوال باستخدام Java: تطوير برمجيات للأجهزة الصغيرة باستخدام J2ME**. حلب: شعاع للنشر والتوزيع، ٢٠٠٥.
- مرجان، هيثم: **Java 2 دورة في كتاب**. حلب: شعاع للنشر والتوزيع، ٢٠٠١.

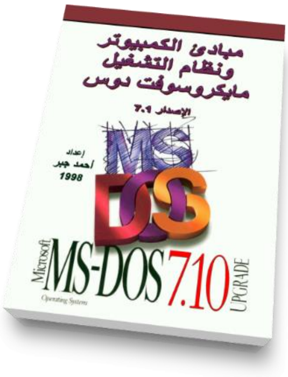
- مركز التعريب والبرمجة (ترجمة): **++C الدليل الكامل: دروس إضافية على الوب مع شهادة علمية**. بيروت: الدار العربية للعلوم، ١٩٩٧.
- موريس، بروس: **تعلم الجافا الآن**. ترجمة: مركز التعريب والبرمجة. بيروت: الدار العربية للعلوم، ١٩٩٦.
- نخبة من المبرمجين: **تقنيات وحلول في JAVA**، جزئين. حلب: شعاع للنشر والتوزيع، ٢٠٠٥.
- النقري، مهيب: **تعرف على البرمجة غرضية التوجه**. دمشق: دار الرضا للنشر، ٢٠٠١.
- النقري، مهيب: **تعرف على لغة JAVA**. دمشق: دار الرضا للنشر، ٢٠٠٠.
- هيوبارد، جون: **ملخصات إيزي شوم: البرمجة باستخدام ++C**. ترجمة: محمد إبراهيم العدوي وهشام عبد المنعم كشك. القاهرة: الدار الدولية للاستثمارات الثقافية ش.م.م، ٢٠٠١.
- هيوبارد، جون: **ملخصات إيزي شوم: البرمجة بلغة JAVA**. تحرير الموجز: أنطوني باكستر، ترجمة: فؤاد شديد زايد. القاهرة: الدار الدولية للاستثمارات الثقافية ش.م.م، ٢٠٠٤.
- وولش، آرون وكاوتش، جوستين وستينبرج، دانيال: **جافا ٢ بايبل JAVA 2 BIBLE**. ترجمة: قسم الترجمة بدار الفاروق وخالد العامري. القاهرة: دار الفاروق للنشر والتوزيع، ٢٠٠١.

ثانياً: أهم المصادر والمراجع الأجنبية:

- Deitel, Harvey M. and Deitel, Paul J.: **JAVA HOW TO PROGRAM**, 5th ed. India: Pearson Education Pte. Ltd, 2004.
- Gosling, B. and Others: **THE JAVA LANGUAGE SPECIFICATION: SECOND EDITION**. Web site: Java.sun.com.
- Jaworski, Jamie: **TRAINING GUIDE: JAVA 2 CERTIFICATION**, 1st ed. India: Techmedia, 1999.
- SYBEX: **JAVA 2, J2SE 1.4 COMPLETE**, 1st ed. India: BPB Publications, 2003.

مبادئ الكمبيوتر
ونظام التشغيل مايكروسوفت دوس MS-DOS
الإصدار ٧،١

(١٩٩٨)



- أســـس ومبادئ الكمبيوتر، والتعريف بأجزائه ومكوناته، وطريقة عمله.
- أسـاسـيات نظم تشـغيل الكمبيوتر، وتعريف بأهم أنظمة التشغيل وأنواعها المختلفة، وفكرة عملها.
- التعرف على نظام التشغيل مايكروسوفت دوس MS-DOS الإصدار ٧،١.
- سطر الأوامر وطرق التعامل معه.
- أهم الأوامر الداخلية في الدوس.
- التعامل مع الملفات والأدلة والأقراص.
- التعامل مع النصوص والملفات النصية.
- التعامل مع الأوامر الخارجية والبرامج التطبيقية.
- إدارة نظام التشغيل وصيانته.
- بعض المهام المتقدمة.

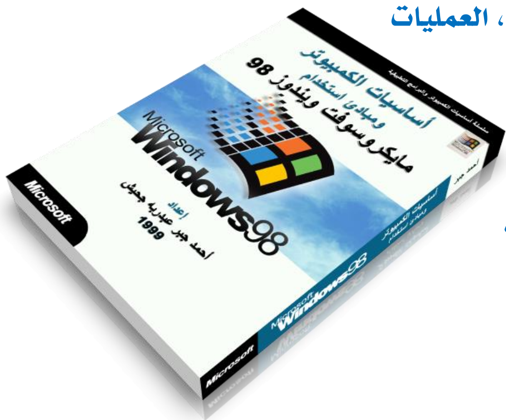


أساسيات الكمبيوتر
ومبادئ استخدام مايكروسوفت ويندوز ٩٨
Microsoft Windows 98

(١٩٩٩)



- أساسيات الكمبيوتر: ما هو الكمبيوتر، المكونات الأساسية للكمبيوتر: المكونات المادية، المكونات الغير مادية.
- نظام التشغيل ويندوز Windows 98: تعريفه، تاريخه، مميزاته.
- مكونات النظام الأساسية والتعامل معها: سطح المكتب، قائمة ابدأ، النوافذ، القوائم، مربعات الحوار.
- إدارة الملفات والمجلدات والأقراص: برنامج مستكشف ويندوز، العمليات الرئيسية على الملفات، سلة المحذوفات، الاختصارات، العمليات الرئيسية على المجلدات، العمليات الرئيسية على الأقراص.
- التعامل مع التطبيقات: الرسام، المفكرة، معالج الكلمات، الآلة الحاسبة، الألعاب.



أساسيات شبكات الكمبيوتر Networks
ومقدمة إلى شبكة الإنترنت Internet

(٢٠٠٠)

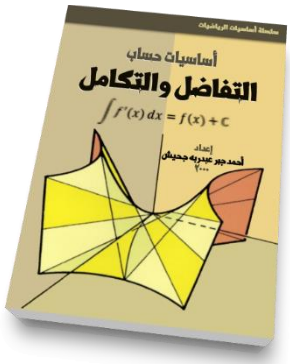


- مدخل إلى شبكات الكمبيوتر Network: تعريفها، أهميتها، معداتها، بروتوكولاتها.
- أنواع الشبكات: أشهر التصنيفات، أهم الأنواع.
- شبكة الإنترنت Internet: ماهيتها، تاريخها، أهميتها وفوائدها.
- خدمات الإنترنت نظرة شاملة.
- شبكة الوب Web: عناوين URL، صفحات Web، مستعرضات Web.
- البريد الإلكتروني E-Mail: عناوين البريد، ارسال واستقبال البريد، إدارة البريد.
- التواصل الآني والمحادثات: خدمة IRC، برنامج ICQ.
- أشهر مواقع الوب العربية في شبكة الإنترنت.



أساسيات حساب التفاضل والتكامل Calculus

(٢٠٠٠)



- **المفاهيم الأساسية:**

  المجموعات، الأعداد الحقيقية، الفترات، المتراجحات، القيم المطلقة.

- **الدوال:**

  المفهوم، الأنواع، دوال كثيرة الحدود، الدوال الأسية، الدوال المثلثية، الدوال الزائدية، تركيب الدوال، الدالة العكسية، الدوال اللوغاريتمية، الدوال المثلثية العكسية، الدوال الزائدية العكسية.

- **النهايات:**

  المفهوم، نهاية الدوال، نظريات النهايات، النهايات غير المعرفة، النهايات عند اللانهاية.

- **الاتصال:**

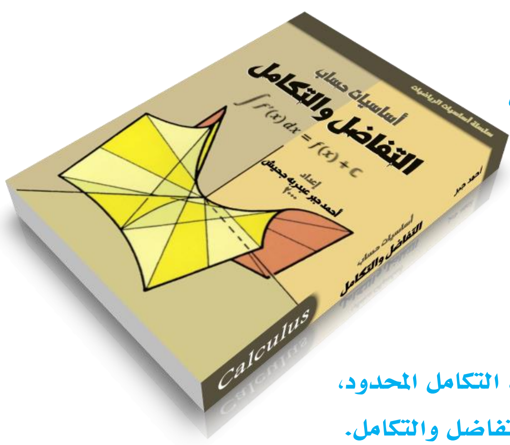
  المفهوم، اتصال الدوال، نظريات الاتصال.

- **المشتقات:**

  المفهوم، قابلية الاشتقاق، قوانين الاشتقاق الأساسية، اشتقاق الدوال المختلفة، الاشتقاق الضمني، التفاضلات، المشتقات النونية.

- **تطبيقات المشتقات:**

  نظرية القيمة المتوسطة، التزايد والتناقص للدالة، التقعر وذقاط الانقلاب، القيم العظمى والصغرى للدالة، رسم منحنى الدالة.



- **التكامل غير المحدود:**

  الدالة الأصلية والمشتقة العكسية، خواص التكامل غير المحدود، التكامل بالتعويض، التكامل بالتجزيء، تكاملات الدوال المثلثية والزائدية، التكامل بالكسور الجزئية، بعض التكاملات الشائعة.

- **التكامل المحدود:**

  المجاميع، حساب المساحات تحت منحنى الدالة، التكامل المحدود، خواص التكامل المحدود، النظرية الأساسية للتفاضل والتكامل.

- **تطبيقات التكامل:**

  المساحات، الحجوم، الشغل.

أساسيات معالجة النصوص
واستخدام مايكروسوفت وورد ٢٠٠٠
Microsoft Word 2000

(٢٠٠١)



- **برامج معالجة النصوص:**

المفاهيم الأساسية، نظرة تاريخية، أهميتها وفوائدها، أشهر برامج معالجة النصوص.

- **البدء مع برنامج مايكروسوفت وورد ٢٠٠٠:**

التعريف، أهم المميزات، فوائد الاستخدام، تشغيل البرنامج، التعرف على النافذة الرئيسية، الخروج من البرنامج.

- **المهام الأساسية للمستندات:**

الكتابة (عربي، إنجليزي)، التعامل مع الملفات (المستندات): الحفظ، الإغلاق، الفتح، إضافة معلومات، حفظ نسخة، الفتح المتعدد، التحكم في عرض النصوص، عرض إحصائيات المستند.

- **التعامل مع النصوص:**

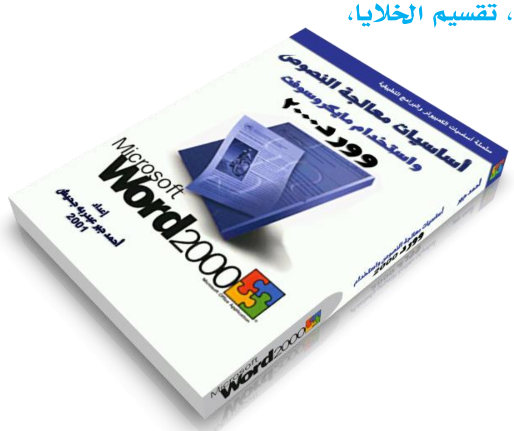
التحرك داخل النص، تحديد أجزاء النص، تنقيح النص.

- **تنسيق النصوص:**

ما قبل التنسيق، تنسيق الفقرات، تنسيق الخطوط، تنسيق الصفحة، مهارات تنسيقية إضافية.

- **التعامل مع الجداول:**

طرق التصميم، التنقل، الكتابة، تحديد أجزاء الجدول، الحذف والإضافة (صف – عمود)، تغيير عرض وارتفاع الأعمدة والصفوف، دمج الخلايا، تقسيم الخلايا، تقسيم الجدول، تنسيق الجدول.



- **بعض المهارات المتقدمة:**

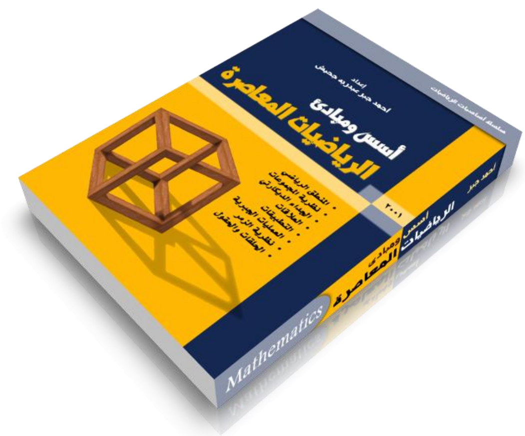
مهارات الكتابة، التعامل مع الصور والرسومات والتأثيرات، مهارات الطباعة، دمج المراسلات.

أسس ومبادئ الرياضيات المعاصرة:
مقدمة في الجبر المجرد

(٢٠٠١)



- المنطق الرياضي.
- نظرية المجموعات.
- الجداء الديكارتي والعلاقات.
- التطبيقات.
- العمليات الجبرية.
- نظرية الزمر.
- الحلقات والحقول.



أساسيات جداول البيانات الإلكترونية
واستخدام برنامج مايكروسوفت إكسل إكس بي
Microsoft Excel XP

(٢٠٠٢)



- **برامج جداول البيانات الإلكترونية:**

المفاهيم الأساسية، نظرة تاريخية، أهميتها وفوائدها، أشهر برامج جداول البيانات الإلكترونية، إجراء الحسابات الإلكترونية.

- **البدء مع برنامج مايكروسوفت إكسل إكس بي:**

التعريف، أهم المميزات، فوائد الاستخدام، تشغيل البرنامج، التعرف على النافذة الرئيسية، الخروج من البرنامج.

- **المكونات الأساسية للمصنف (ملف إكسل):**

المصنف (دفتر العمل)، الورقة، الأعمدة، الصفوف، الخلايا، تخطيط المسودة، إدخال العناوين، النصوص، الأرقام.

- **المهام الأساسية للمصنفات:**

الحفظ، إضافة المعلومات، إنشاء مصنف عمل جديد، فتح مصنف عمل قديم، العمل باللغة الإنجليزية، عرض أكثر من مصنف عمل معاً.

- **التعامل مع أوراق العمل:**

تسمية ورقة العمل، التعامل مع بطاقات أوراق المصنف، إدارة الأوراق: إضافة – حذف – نقل – نسخ، تحديد جزء من ورقة العمل (خلية – خلايا متجاورة – خلايا متفرقة – نطاق من الخلايا – صف كامل – عمود كامل – كل خلايا الورقة).

- **التعامل مع منطقة البيانات:**

الخلايا (إضافة – حذف – تسمية)، الصف (إضافة – حذف)، العمود (إضافة – حذف)، تغيير ارتفاع صف أو عدة صفوف، تغيير عرض عمود أو عدة أعمدة، استعادة العرض الافتراضي.

- **تعديل محتويات الخلايا:**

استبدال محتويات خلية، إصلاح الأخطاء داخل الخلية، الإضافة إلى محتويات الخلية، حذف محتويات خلية أو أكثر، التراجع عن آخر تعديل، تكرار آخر وظيفة، نسخ محتويات خلية، النسخ التلقائي في الخلايا المتجاورة، نقل محتويات خلية، تحريك نطاق من الخلايا، البحث والاستبدال، فرز وترتيب محتويات الخلايا.

- **العمليات الحسابية:**

الجمع والطرح، الضـرب والقسـمة والنسـبة، ضـم النصـوص، مقارنة القيم، مقدمة عن اسـتخدام الوظيفة الشرطية IF، أولويات الحساب، تغيير أولويات الحساب.

- **المعادلات الحسابية والدوال الجاهزة:**

اسـتخدام عناوين الخلايا في المعادلات الحسـابية، اسـتخدام أسماء الخلايا في المعادلات الحسـابية، اسـتخدام الدوال الجاهزة، الجمع التلقائي، النسـخ التلقائي للمعادلات، اسـتخدام العناوين المطلقة، استخدام معالج إنشاء الوظائف.

- **أهم الدوال (الوظائف) الجاهزة:**

الدالة AVERAG، الدالة COUNT، الدالة IF، الدالة MAX، الدالة MIN، الدالة NOW، الدالة ROUND، أهم الدوال الرياضية والمثلثية.

- **التنسيق في ورقة العمل:**

المحاذاة داخل الخلايا، تغيير اتجاه النص في الخلية، تنسـيق الخط، تنسـيق الأرقام (كنسـبة – علامة عشـرية – خانات عشـرية – عملة – لون عرض الأرقام السـالبة)، تخطيط خلايا الجدول، تغيير لون الخلايا، تغيير لون الخطوط، تظليل الخلايا بالنقوش، إخفاء خطوط الشـبكة، تنسـيق الصفحات.

- **الرسوم البيانية:**

إنشاء الرسم في ورقة خاصة، إنشاء الرسم في نفس الورقة، استخدام معالج الرسوم، تعديل مكونات الرسم البياني، الرسوم البيانية الثلاثية الأبعاد، تنسيق الرسوم البيانية.

- **قاعدة البيانات:**

مقدمة عن قواعد البيانات، مفهوم الحقول والسـجلات، تصـميم قاعدة البيانات، إدخال السـجلات، استخدام نموج البيانات، الانتقال بين السجلات، تعديل بيانات سجل، التراجع عن تعديل السجل، حذف سـجل، فرز وترتيب السـجلات، البحث في قاعدة البيانات، البحث باستخدام الفلتر التلقائي.

- **الطباعة:**

طباعة محتويات مصـنف، تحديد مقاس وإعدادات الصفحة، تعديل هوامش الصفحة، موضـع الجدول في الصـفحة، تصـميم رأس وتذييل الصـفحة، طباعة خطوط الشـبكة، معاينة الصـفحة، طباعة جزء من الورقة، طباعة الرسم البياني فقط.



مقدمة في الجبر الخطي Linear Algebra:
أسس ومبادئ نظرية المصفوفات Matrix والمحددات Determinant

(٢٠٠٢)



- **المصفوفات:**

التعريف، أنواع خاصة من المصفوفات، المصفوفة الصفرية، مصفوفة الوحدة.

- **جبر المصفوفات:**

تساوي المصفوفات، جمع المصفوفات، النظير الجمعي، المحايد الجمعي، ضرب المصفوفة بعدد، طرح المصفوفات، خصائص عملية الجمع، ضرب المصفوفات، النظير الضربي، المحايد الضربي، خصائص عملية الضرب.

- **عمليات خاصة على المصفوفات:**

مدور المصفوفة، أثر المصفوفة، العمليات الصفية الأولية، المصفوفات الأولية، معكوس المصفوفة المربعة، قابلية الانعكاس، تشابه المصفوفات، رفع المصفوفة إلى قوة.



- **المحددات:**

التعريف، نشر المحدد، طريقة ساروس لحساب المحددات، محددات المصفوفات القطرية أو المثلثية.

- **العمليات على المحددات:**

محدد المدور، تأثير العمليات الصفية/العمودية الأولية على المحددات، مصفوفات محدداتها أصفار، خواص حساب المحددات، محددات بدلالة متغير، الاستفادة من المحددات في إيجاد معكوس المصفوفات.

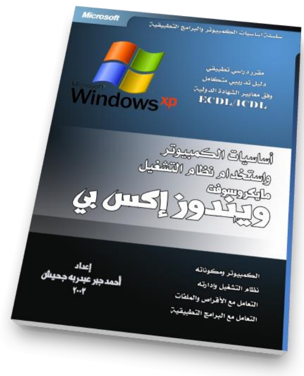
- **تطبيقات على المصفوفات والمحددات:**

حل نظام المعادلات الخطية باستخدام المصفوفات.

أساسيات الكمبيوتر
واستخدام نظام التشغيل مايكروسوفت ويندوز إكس بي
Microsoft Windows XP

(٢٠٠٣)

مقرر دراسي تطبيقي، دليل تدريبي متكامل وفق معايير الشهادة الدولية ECDL/ICDL



- **أساسيات الكمبيوتر:**

ما هو الكمبيوتر، تاريخه وتطوره، أنواع الكمبيوترات.

- **المكونات الأساسية للكمبيوتر:**

المكونات المادية (جسد الكمبيوتر): وحدات الإدخال (Input Unit)، وحدات الإخراج (Output Unit)، وحدة المعالجة المركزية (Center Processing Unit - CPU)، وحدة الذاكرة الرئيسية، وحدات قياس السعة التخزينية لذاكرة الكمبيوتر، وحدات التخزين الثانوية - الدائمة: الأقراص الصلبة، الأقراص المرنة، الأقراص الليزرية المدمجة،

المكونات غير المادية (روح الكمبيوتر): أنظمة التشغيل Operating Systems ، البرامج التطبيقية Application Programs.

- **نظام التشغيل ويندوز Windows:**

ما هو نظام التشغيل ويندوز، مميزات ويندوز.

- **أهم مكونات النظام ويندوز:**

سطح المكتب Desktop، مكوناته، شريط المهام، مكوناته، زر ابدأ، مكوناته، قائمة البرامج، كيفية تشغيل البرامج، قائمة المستندات، فتح آخر المستندات، قائمة إعدادات، أمر البحث، أمر تعليمات، أمر تشغيل، أمر إيقاف تشغيل النظام.

- **التعامل مع واجهات النظام ومكوناته:**

التعامل مع الفأرة (الماوس)، التعامل مع النوافذ، مكوناتها، التحجيم والتحريك، التعامل مع القوائم، القوائم الفرعية، القائمة السريعة، التعامل مع مربعات الحوار، مربعات الحوار المبوبة، التعامل مع الاختصارات.

- **جهاز الكمبيوتر، مكوناته:**

الأقراص الثابتة، الأقراص المرنة، الأقراص المضغوطة.

- **المستكشف Explorer:**

طرق تشغيل المستكشف، مكوناته، طرق العرض، الترتيب، خيارات عرض المجلد، استخدام لوحة المفاتيح مع المستكشف.

- **التعامل مع الملفات والمجلدات:**

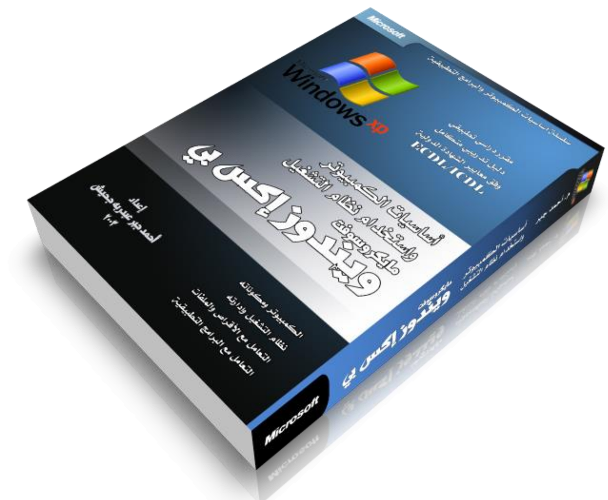
رموز عرض المجلدات والملفات والأنواع الأساسية للملفات، إنشاء مجلد أو ملف، الأسماء الطويلة للملفات، تحديد المجلدات والملفات، إعادة تسمية مجلد أو ملف، نسخ الملفات والمجلدات، نقل الملفات والمجلدات، السحب والإفلات، للنسخ والنقل، إرسال إلى ....، التراجع عن العمليات التي قمت بها، إلغاء عمليات الملفات أثناء القيام بها، خصائص المجلدات والملفات وتغييرها، حذف الملفات والمجلدات، سلة المحذوفات، استعادة الملفات المحذوفة.

- **لوحة التحكم بالنظام:**

خصائص عرض النظام، إعدادات شريط المهام وقائمة ابدأ، إعدادات الفأرة-الماوس، إعدادات التاريخ والوقت، الإعدادات الإقليمية.

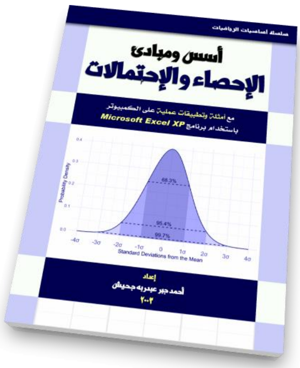
- **تشغيل البرامج التطبيقية:**

برنامج الدفتر، برنامج الرسام، الحاسبة.



أسس ومبادئ الإحصاء Statistics
والاحتمالات Probability

(٢٠٠٣)



- **الإحصاء وعرض البيانات:**

  طرق عرض البيانات، التوزيعات التكرارية، التكرارات النسبية، التوزيع التكراري المتجمع، طرق تمثيل التوزيعات التكرارية بيانياً، حالات المنحنيات التكرارية، تطبيقات على برنامج إكسل.

- **مقاييس النزعة المركزية:**

  الوسط الحسابي وخواصه، الوسيط وتطبيقاته، المنوال وحالاته، العلاقة بين الوسط والوسيط والمنوال، الربيعات والعشيرات والمئويات، الوسط الهندسي والتوافقي والتربيعي، تطبيقات على برنامج إكسل.

- **مقاييس التشتت والالتواء والتفرطح:**

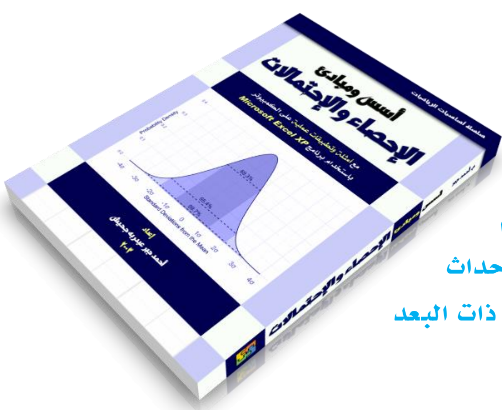
  المدى، المدى الربيعي، الانحراف الربيعي، الانحراف المطلق، التباين وخواصه، الانحراف المعياري، معامل الاختلاف، العزوم والالتواء والتفرطح، تطبيقات على برنامج إكسل.

- **طرق جمع البيانات والعينات:**

  المعاينة وإطارها، أنواع طرق الجمع، العينات وأنواعها الاحتمالية (العشوائية البسيطة، المنتظمة، الطبقية، متعددة المراحل، العنقودية، المحاصصة، المعيارية) وغير الاحتمالية (الملائمة، التمييز).

- **نظرية المجموعات وطرق العد:**

  المجموعات وأنواعها، جبر المجموعات، خواص العمليات الجبرية، طرق العد، التباديل وتطبيقاته، التوافيق وتطبيقاته.



- **نظرية الاحتمالات:**

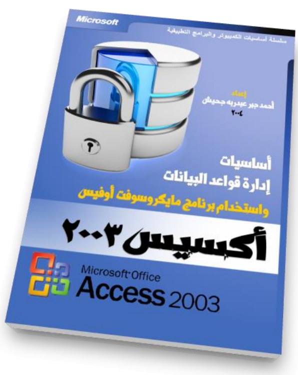
  التجربة العشوائية، فضاء العينة، الحوادث (المستحيلة، المؤكدة، المتنافية)، الاحتمال، مسلمات وقوانين الاحتمالات، الاحتمال المنتظم والتكرار النسبي، الاحتمال الشرطي، الأحداث المستقلة، قاعدة بيز، المتغيرات العشوائية ذات البعد الواحد، التوزيعات الاحتمالية.

- **الارتباط والانحدار:**

الارتباط، معامل الارتباط الخطي البسيط (معامل ارتباط بيرسون)، معامل ارتباط الرتب (معامل ارتباط سبيرمان للرتب)، الانحدار، أشكال الانتشار، تحليل الانحدار، توفيق الانحدار، الانحدار المتعدد، تطبيقات على برنامج إكسل.

أساسيات إدارة قواعد البيانات
واستخدام برنامج مايكروسوفت أوفيس أكسيس ٢٠٠٣
Microsoft Office Access 2003

(٢٠٠٤)



- **مدخل إلى قواعد البيانات Databases:**

المفاهيم الأساسية، نظرة تاريخية، أهمية وفوائد قواعد البيانات، أنواع قواعد البيانات.

- **نظم إدارة قواعد البيانات DBMS:**

ماهيتها، تطورها، أهميتها وفوائدها، أشهرها.

- **قواعد البيانات العلائقية Relational Database:**

النموذج العلائقي، أنواع العلاقات، الجبر العلائقي وعملياته، مراحل تصميم وبناء قاعدة بيانات علائقية، النماذج المعيارية Normal Forms.

- **برنامج مايكروسوفت أوفيس أكسيس ٢٠٠٣:**

التعريف، أهم المميزات، التعامل مع البرنامج، إنشاء قاعدة بيانات وإدارتها.

- **التعامل مع الجداول Tables:**

التعريف، طرق الإنشاء، إدخال البيانات، إدارة الجداول، التصميم اليدوي، أنواع البيانات، إدارة الحقول، إدارة السجلات، طرق العرض، البحث والفرز والتصفية.



- **بناء العلاقات Relationships بين الجداول:**

أهميتها، أنواعها في أكسيس وأين ومتى تستخدم، إنشاء علاقة بين جدولين، التكامل المرجعي، إدارة العلاقات.

- **التعامل مع النماذج Forms:**

التعريف، الأهمية والفوائد، طرق الإنشاء، إدخال البيانات والتنقل بين الحقول والسجلات، التعامل مع البيانات في النموذج: البحث والفرز والتصفية، إدارة النماذج، التصميم اليدوي.

- **التعامل مع الاستعلامات Queries:**

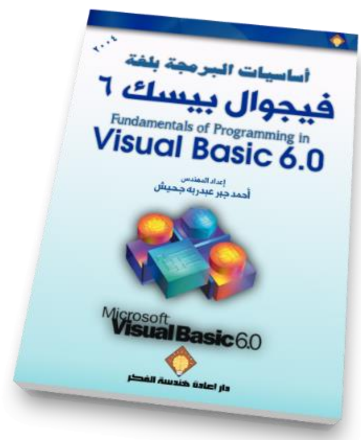
التعريف، الأنواع، الأهمية والفوائد، طرق الإنشاء، التصميم اليدوي، نظرة سريعة على لغة SQL، التعامل مع البيانات في الاستعلام: البحث والفرز والتصفية، إدارة الاستعلامات.

- **التعامل مع التقارير Reports:**

التعريف، الأهمية والفوائد، طرق الإنشاء، التصميم اليدوي، إدارة التقارير، معاينة وطباعة التقرير.

أساسيات البرمجة بلغة فيجوال بيسك ٦
Fundamentals of Programming in Visual Basic 6.0

(٢٠٠٤)



- **مقدمة عن البرمجة والبرمجة المرئية:**

مقدمة تاريخية عن لغة البيسيك وفيجوال بيسيك، مفهوم البرمجة المرئية، مفهوم البرمجة المسيرة بالأحداث Event-Driven Programming، مفهوم العناصر والكائنات Objects and Controls، البرمجة الموجهة بالكائنات Object-Oriented Programming (OOP).

- **البدء مع بيئة فيجوال بيسيك:**

تشغيل فيجوال بيسيك، بيئة التطوير المتكاملة IDE للغة فيجوال بيسيك، مفهوم المشروع – النموذج (الإنشاء – الفتح) – أنواع الملفات، مربع الأدوات – إضافة الأدوات وتغيير موقعها وحجمها، إطار الخصائص – تغيير خاصية، إطار المشروع.

- **خطوات البرمجة المرئية:**

كتابة البرنامج الأول، إنشاء واجهة البرنامج، ضبط الخصائص، كتابة شيفرة البرنامج – العبارات البرمجية – إنشاء الأحداث، حفظ البرنامج، تشغيل البرنامج، بناء الملف التنفيذي.

- **العمل مع أدوات التحكم:**

الاستعمال الأساسي لأدوات التحكم (زر الأمر – مربع النص – العنوان أو التسمية)، استعمال كائنات نظام الملفات (الأقراص – المجلدات – الملفات)، كائن الصورة – برنامج مستكشف الصور، كائنات للحصول على إدخالات المستخدم (زر خيار – مربع اختيار)، التعامل مع مربع السرد والقوائم المركبة المنسدلة، العمل مع القوائم – محرر القوائم، التعامل مع مربعات الحوار (OPEN، SAVE AS, PRINT, FONT, COLOR).

- **أساسيات البرمجة بلغة فيجوال بيسيك الجزء الأول:**
- **المتغيرات والعوامل في فيجوال بيسيك:**

هيكلية جمل فيجوال بيسيك، استعمال المتغيرات لتخزين المعلومات، حجز مساحة للمتغيرات (عبارة DIM)، تعيين القيمة للمتغير، تصريح المتغيرات من دون استعمال DIM، الإجبار على التصريح بعبارة OPTION EXPLICIT، الحصول على إدخال من المستخدم بواسطة الدالة INPUTBOX، عرض الرسائل والإخراج في مربع حوار بواسطة MSGBOX.

- **أنواع البيانات في فيجوال بيسك:**

أنواع البيانات الأساسية، أنواع البيانات المعرفة من قبل المستخدم (استخدام العبارة TYPE)، الثوابت (التعريف والاستخدام)، استعمال عوامل فيجوال بيسيك (+،-،*،/،\،mod،^،&)، تطبيق الآلة الحاسبة البسيطة، دوال فيجوال بيسيك الرياضية (abs, cos, exp, rnd, sgn, sin, sqr, str, val)، أولويات العوامل.

- **استعمال بنيات القرار:**

استعمال التعابير الشرطية – عوامل المقارنة (=، <>، <، >، <=، >=)، بنيات القرار IF...THEN، استعمال العوامل المنطقية في التعابير الشرطية (AND, OR, NOT, XOR)، بنية القرار SELECT CASE، استعمال عوامل المقارنة مع البنية SELECT CASE واستخدام IS,TO.

- **استعمال الحلقات وعدادات الوقت:**

حلقات FOR...NEXT، الجملة EXIT FOR، حلقات DO WHILE LOOP و DO LOOP WHILE وDO LOOP UNTIL، استعمال كائنات عدادات الوقت.

- **العمل مع النماذج:**

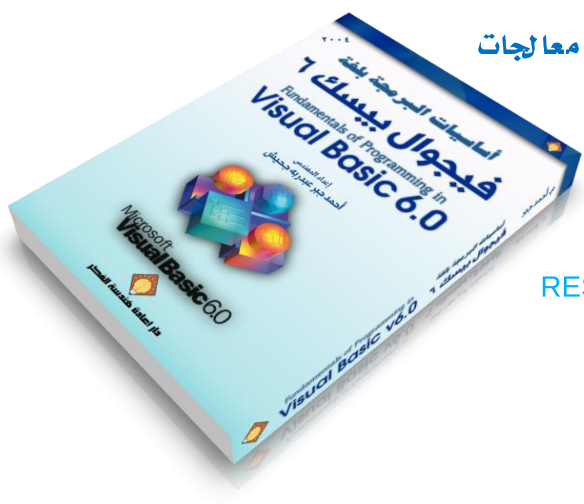
إضافة نماذج جديدة إلى البرنامج (النماذج الفارغة، النماذج الموجودة سابقاً، النماذج الجاهزة مسبقة التصميم – الشاشة الافتتاحية، إرشادات البرنامج، مربعات الحوار)، جمل وعبارات النموذج في الشيفرة البرمجية (Load، Show، Hide، Unload)، خصائص النموذج، العمل مع المشاريع متعددة النماذج، الواجهات متعددة الوثائق MDI، تطبيق المفكرة متعددة النماذج.

- **الطباعة في فيجوال بيسك:**

إرسال خرج البرنامج إلى الطابعة، الكائن PRINTER، طباعة نموذج بأكمله باستعمال الطريقة PRINTFORM.

- **التعامل مع الأخطاء باستعمال معالجات الأخطاء:**

متى يجب استخدام معالجات الأخطاء، إعداد مصائد الأخطاء – الجملة ON ERROR GOTO، الجملة RESUME – RESUME NEXT.



- **إضافة الأعمال الفنية والتأثيرات الخاصة:**

أداة التحكم خط، أداة التحكم شكل، تغيير مؤشر الفأرة الخصائص – MOUSEPOINTER MOUSEICON، نقل الكائنات في نظام الإحداثيات – الطريقة MOVE.

- **أساسيات البرمجة بلغة فيجوال بيسيك الجزء الثاني:**
- **العمل مع الوحدات النمطية:**

إنشاء وحدة نمطية قياسية، العمل مع المتغيرات العمومية PUBLIC.

- **العمل مع الإجراءات:**

إنشاء الإجراءات الفرعية SUB، تمرير المتغيرات بالمرجع وبالقيمة استخدام – BYREF BYVAL، استدعاء الإجراءات الفرعية.

- **العمل مع الدوال:**

إنشاء الدوال FUNCTION، استدعاء الدوال، قطع عمل الإجراءات أو الدوال والعبارة EXIT، أهم دوال معالجة السلاسل النصية (right, left, mid, instr, asc, chr ،ucase, lcase, len).

- **العمل مع المصفوفات:**

مفهوم المصفوفات وأنواعها (مصفوفات ثابتة الحجم – مصفوفات ديناميكية)، إنشاء المصفوفة ثابتة الحجم، العمل مع عناصر المصفوفة، إنشاء المصفوفة الديناميكية، استخدام جملة REDIM لتغيير حجم المصفوفة، المصفوفات متعددة الأبعاد، مصفوفة العناصر ActiveX، التعامل مع مصفوفة العناصر والخاصية INDEX.

- **برمجة قواعد البيانات بواسطة فيجوال بيسيك:**

إنشاء تطبيقات مخصصة لقواعد البيانات، مناهج برمجة قواعد البيانات (DAO – RDO – ADO)،

استعمال الأداة DAO عنصر التحكم بالبيانات وخصائصها (,CONNECT DATABASENAME, RECORDSETTYPE ،READONLY ،RECORDSOURCE)،

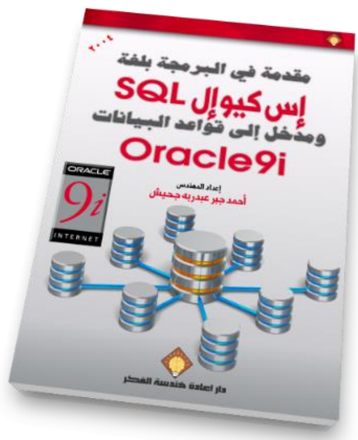
استعمال أدوات التحكم المرتبطة bound لعرض البيانات من القاعدة، الخاصيتين DATASOURCE – DATAFIELD،

استعمال كائن مجموعة سجلات RECORDSET، الإضافة والحذف للسجلات، التحقق من البيانات المدخلة،

استخدام كائنات البيانات ActiveX Data Objects ADO، خصائص ADO الخاصة، مقدمة للبرمجة باستخدام مكتبة ADODB.

مقدمة في البرمجة بلغة إس كيو إل SQL
ومدخل إلى قواعد البيانات Oracle 9i

(٢٠٠٤)



- **بعض المفاهيم الأساسية:**

البيانات، المعلومات، قاعدة البيانات، نظم إدارة قواعد البيانات (DBMS)، وظائف قواعد البيانات، مفاهيم قاعدة البيانات العلائقية، المستخدمين والمهام، الأعمدة، الصفوف، العلاقات، مفهوم الكائنات.

- **المبادئ الأساسية للغة SQL:**

مفهوم اللغات المستضافة، لغة SQL ومكوناتها (DML, DDL, DCL, TCL)، استخدام برنامج أكسس Access في تنفيذ عبارات لغة SQL.

- **أوامر لغة SQL المستخدمة في الاختيار والاستفسار:**

أوامر اختيار مجموعة من الحقول من أكثر من جدول (SELECT – FROM)، استخدام الشرط WHERE مع الأمر SELECT، استخدام معايير التصفية (معاملات التشغيل: المعامل LIKE والرموز (_، %)، المعامل IN، المعامل BETWEEN)، استخدام عمليات الربط المنطقية (AND, OR)، استبعاد التكرارات في بيانات حقل معين DISTINCTROW، تسمية الحقل المستنتج داخل أمر SELECT، اختيار مجموعة من الحقول من أكثر من جدول، ترتيب السجلات ORDER BY، استخدام الدوال (AVG, COUNT, MIN, MAX, SUM, FIRST, LAST) مع جملة SELECT، الاستفسارات المتداخلة، إنشاء الاستعلامات المعقدة بواسطة برنامج ACCESS.

- **مقدمة عن أوراكل وبيئة العمل فيها:**

نظرة عامة على ORACLE، تعريف قاعدة البيانات ORACLE، البرامج المساعدة في أوراكل (برنامج SQL*PLUS، برنامج Enterprise Manager)، العناصر الرئيسية الخاصة بقاعدة بيانات أوراكل (عناصر Tablespaces، الجداول، عناصر Schemas، عناصر Sequence، الإجراءات المخزنة والوظائف ومجموعة الإجراءات، الأزندة (القوادح) TRIGGERS، المرادفات (SYNONYMS)، مقدمة عن استخدام SQL*PLUS (تشغيل البرنامج، سطر الأوامر الخاص بـ SQL*PLUS، الاتصال بقاعدة البيانات باستخدام SQL*PLUS، المحث (SQL>)، إدخال الأوامر والفاصلة المنقوطة).

- **لغة معالجة البيانات DML (التعامل مع البيانات):**

أمر إضافة السجلات INSERT INTO – VALUES (إضافة سجل واحد، نسخ السجلات من جدول إلى آخر، استخدام طريقة & في الإضافة)، أمر تحديث البيانات UPDATE – SET (تحديث الحقول في سجلات معينة، تحديث الحقول في كافة السجلات)، أمر حذف السجلات DELETE FROM (حذف سجلات معينة، حذف كافة سجلات الجدول)، أمر إبطال التعديلات ROLLBACK، حفظ التعديلات COMMIT.

- **لغة تعريف البيانات DDL (التعامل مع الكائنات):**

التعامل مع الجداول TABLES (إنشاء جدول، إنشاء جدول فارغ، إنشاء جدول من خلال جدول أخر، الاستعلام عن بنية وتكوين الجدول DESC، تعديل خصائص جدول، حذف جدول، الاستعلام عن الجداول الموجودة)، القيود على الجداول CONSTRAINTS (إضافة القيود على مستوى العمود، على مستوى الجدول (استخدام العبارة CONSTRAINT)، أنواع القيود (عدم قبول القيم الفارغة NOT NULL، عدم التكرار UNIQUE، شرط التحقق CHECK، المفتاح الرئيسي PRIMARY KEY، المفتاح الأجنبي (بناء العلاقة) FOREIGN KEY، إضافة قيد بعد إنشاء الجدول، حذف القيود، حذف قيد المفتاح الرئيسي، الاستعلام عن قيود الجداول من خلال الجدول USER_CONSTRAINTS.

- **التعامل مع العروض VIEWS:**

تعريف العروض واستخداماتها، إنشاء العروض، حذف العروض، الاستعلام عن العروض الموجودة بواسطة USER_VIEWS.

- **التعامل مع المتسلسلات SEQUENCES:**

تعريف المتسلسلات، توليد الأرقام المتسلسلة تلقائياً، الاستخدام لأكثر من جدول، الاستقلال عن الجدول، إنشاء المتسلسلات، طريقة استخدام المتسلسلات، حذف المتسلسلات، الاستعلام عن المتسلسلات الموجودة بواسطة USER_SEQUENCES.

- **التعامل مع الفهارس INDEXES:**

تعريف الفهارس واستخدامها، تسريع الحصول على البيانات، الاستقلال عن الجدول، حذفها لا يؤثر في الجدول، تحذف عند حذف جدولها، إنشاء فهرس، حذف فهرس (ماعدا الحقول NOT NULL,PRIMARY KEY)، متى تستخدم الفهارس، متى لا تستخدم الفهارس، الاستعلام عن الفهارس الموجودة بواسطة USER_INDEXES.

- **التعامل مع المرادفات SYNONYMS:**

تعريف المرادفات واستخدامها كبدائل لأسماء الجداول والعروض، إنشاء المرادفات، حذف المرادفات، الاستعلام عن المرادفات الموجودة بواسطة USER_SYNONYMS.

- **أهم الأوامر المستخدمة في SQL*PLUS:**

أمر مسح محتويات المخزن المؤقت الخاص بـــ SQL (Clear SQL)، أمر الاتصال بوحدة خدمة (Connect)، أمر تقديم وصف تفصيلي لأحد عناصر قاعدة البيانات (Describe)، أمر تحرير محتويات المخزن المؤقت الخاص بـــ SQL (Edit)، أمر تحميل محتويات ملف إلى المخزن المؤقت (Get)، أمر تقديم التعليمات الأساسية لأوامر SQL*PLUS (Help)، أمر وضع قائمة للجمل الموجودة في المخزن المؤقت (List)، أمر الخروج من SQL*PLUS (Quit)، أمر و ضع قائمة بالجمل الموجودة في المخزن المؤقت وتنفيذها (Run)، أمر حفظ المخزن المؤقت في ملف (Save)، أمر توجيه المخرجات إلى ملف خارجي أو الطابعة (Spool)، أمر تشـــغيل جمل SQL الموجودة بداخل ملف (Start).

- **نظام التأمين في ORACLE – مستخدمي ORACLE:**

اتصال المستخدم بقاعدة البيانات، إنشاء مستخدم جديد، منح الصلاحيات (الامتيازات) للمستخدم، تعريف الأدوار ROLE (صندوق الصلاحيات) وأهميتها، إنشاء الأدوار، إعطاء الامتيازات للأدوار، إعطاء امتيازات الدور للمســـتخدم، امتيازات الكائنات Object Privileges، الاســـتعلام عن الامتيازات الممنوحة للأدوار، الا ستعلام عن الأدوار الممنوحة للمستخدمين، الا ستعلام عن امتيازات الكيانات الممنوحة للمســـتخدمين، ســـحب الامتيازات من المســـتخدم REVOKE، تغيير كلمة المرور للمستخدم، حذف المستخدم.

- **لغة الاستعلامات البنيوية الإجرائية PL/SQL:**

التعامل مع الكتل BLOCKS (تعريفها وفوائدها)، مكونات الكتل (جزء التصريحات DECLARE، الجزء التنفيذي BEGIN…، تحديد الاســـتثناءات (جزء مراقبة الأخطاء) EXCEPTION، التعليقـات)، أنواع الكتـل (الكتـل الغير محـددة (مســـمـاة) ANONYMOUS، الإجراءات PROCEDURES، الدوال FUNCTIONS)، استخدام الكتل، تنفيذ الكتل بواسطة المعامل "/".

- **التعامل مع المتغيرات وأنواعها VARIABLES:**

المتغيرات استخداماتها وفوائدها، أنواع البيانات القياسية (VARCHAR2، NUMBER، DATE، CHAR، LONG، BOOLEAN، BLOB، CLOB، NCLOB، BFILE)، تعريف المتغيرات، تحميل المتغيرات بقيمة، أنواع البيانات المعرفة من قبل المســـتخدم (متغير الســـجل RECORD، اســـتخدام العبارة TYPE%، اســـتخدام العبارة ROWTYPE% لمتغيرات الســـجل، المتغيرات الخارجية (متغيرات SQLPLUS) التعريف – الا سناد – طباعة القيمة، عرض البيانات من داخل الكتلة – تهيئة وحدة الإخراج – اســـتخدام الدمج، اســـتخدام المتغيرات في عبارات SQL، تحميل المتغيرات بقيمة من قاعدة البيانات، استخدام الدوال مع المتغيرات، الثوابت CONSTANT.

- **بنى التحكم واتخاذ القرارات:**

بنية الشرط IF، بنية التكرار LOOP، بنية التكرار FOR، بنية التكرار WHILE.

- **التعامل مع المؤشرات CURSOR:**

المؤشرات وفوائدها وا ستخدامها، تعريف المؤ شرات، معاملات المؤ شرات، فتح المؤ شر، الا سترجاع – تحميل بيانات المؤشر إلى متغير، مواصفات وخصائص المؤشرات، إغلاق المؤشر، استخدام FOR مع المؤشرات.

- **الإجراءات PROCEDURE:**

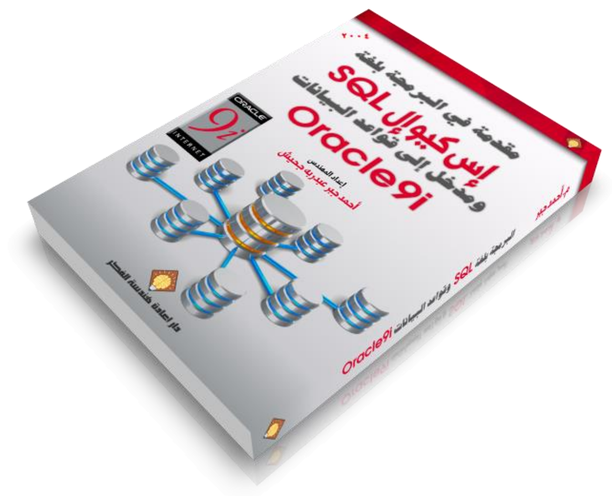
تركيب الإجراءات، إنشـــاء الإجراءات، تنفيذ الإجراءات، تنفيذ الإجراء من داخل إجراء آخر، استخدام المعاملات Parameters، طرق إرسال المعاملات (معاملات IN ، OUT)، حذف الإجراءات.

- **الدوال FUNCTION:**

تركيب الدوال، إنشـــاء الدوال، اســتخدام المعاملات IN في الدوال، تنفيذ الدوال، تنفيذ الدوال من داخل كتل أخرى، حذف الدوال.

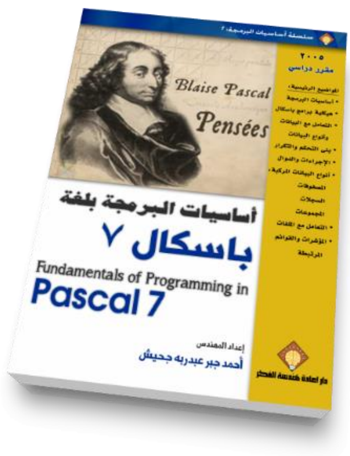
- **الأزندة – القوادح TRIGGERS:**

الأزندة التعريف والتركيب والأنواع، إنشـــاء الأزندة، إيقاف عمل الأزندة وإعادة التشـــغيل، حذف الأزندة.



أساسيات البرمجة بلغة باسكال ٧
Fundamentals of Programming in Pascal 7

(٢٠٠٥)



- **أساسيات البرمجة – مقدمة عن لغات البرمجة:**

الحاسب الآلي ومكوناته، لغات البرمجة (لغة الآلة، لغة التجميع، اللغات عالية المستوى)، نبذة تاريخية عن لغة البرمجة باسكال، لغة باسكال والبرمجة الهيكلية، المكونات الأساسية لبيئة عمل توربو باسكال قياسية.

- **مراحل عمل برنامج بلغة باسكال:**

كتابة البرنامج في المحرر، المعالجة القواعدية – مرحلة ما قبل الترجمة، مرحلة الترجمة، مرحلة الربط، مرحلة التحميل والتنفيذ، مقدمة في البرمجة بلغة باسكال – البرنامج الأول .. طباعة جملة.

- **بنية البرنامج بلغة باسكال:**

رأس البرنامج، قسم التصريحات الاختيارية، الجسم الرئيسي للبرنامج، العبارات البسيطة والعبارات المركبة.

- **التعامل مع البيانات في لغة باسكال:**

المتغيرات (المتحولات)، تسمية المتغيرات – القيود المفروضة على الأسماء، التصريح عن المتغيرات، المعاملات (معامل الإسناد (=:)، المعاملات الحسابية (+و–و*و/)، معاملات المقارنة (<و>و=)، أسبقية المعاملات)، تعريف الثوابت.

- **الإجراءات مسبقة التعريف لعمليتي الإدخال والإخراج:**

الإجراءات المستخدمة في عمليات الإخراج (write, writeln)، الإجراءات المستخدمة في عمليات الإدخال (read, readln).

- **أنواع البيانات القياسية في لغة باسكال:**

الأعداد الحقيقية (نوع البيانات عدد حقيقي (real)، إظهار الأعداد الحقيقية، المعاملات المطبقة على القيم الحقيقية)،

الأعداد الصحيحة (نوع البيانات عدد صحيح (integer)، التحكم بشكل الخرج، نوع البيانات عدد صحيح طويل (LONGINT)، نوع البيانات كلمة (WORD)، نوع البيانات عدد صحيح قصير (SHORTINT)، نوع البيانات بايت (BYTE)، المعاملات المطبقة على الأعداد الصحيحة)،

أنواع البيانات ذات الأساس الرمزي (نوع البيانات رمز (CHAR)، نوع البيانات سلسلة رمزية (STRING)، التحكم بشكل الخرج)،

أنواع البيانات المنطقية (BOOLEAN)، المعاملات المطبقة على المتغيرات المنطقية (AND,OR,NOT)،

أنواع البيانات التعدادية (استخدام التعريف TYPE لإنشاء أنواع البيانات التعدادية، التصريح عن المتغيرات من النوع التعدادي، استخدام المتغيرات من النوع التعدادي).

- **بنى التحكم والتكرار في لغة باسكال:**

بنية الاختيار IF، عبارة IF ELSE، عبارات IF المتداخلة،

الاختيار ما بين عدة إمكانيات وعبارة CASE،

عبارة GOTO والتكرار،

التكرار البسيط – عبارة FOR، الحلقات العكسية والعبارة DOWNTO، حلقات FOR المتداخلة،

بنى التكرار المشروط، بنية التكرار المتهورة REPEAT UNTIL، بنية التكرار الحذرة WHILE.

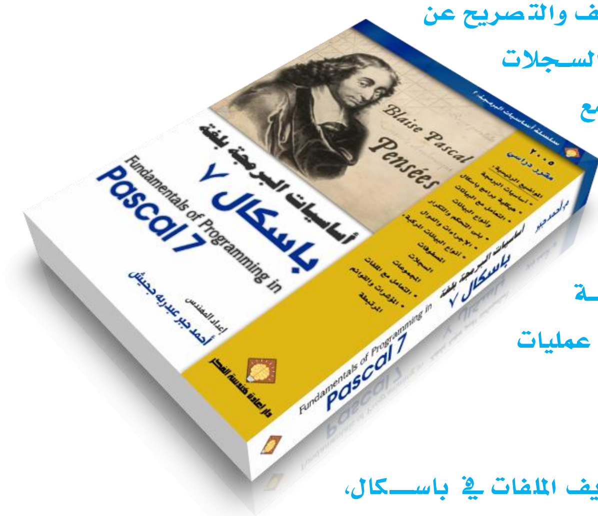
- **الإجراءات والدوال:**

الإجراءات واستخدامها، تعريف الإجراءات، استدعاء الإجراءات، المعاملات (الوسطاء) – المعاملات المرجعية، المتغيرات العامة والمتغيرات المحلية،

الدوال واستخدامها، تعريف الدوال، استدعاء الدوال، الإجراءات والدوال مسبقة التعريف (ABS, SQR, SQRT, INC, DEC, LENGTH, RANDOM, UPCASE).

- **أنواع البيانات المركبة:**

نوع البيانات مصفوفة ARRAY، التعريف والتصريح عن المصفوفة، الكتابة والقراءة من عناصر المصفوفة، المصفوفات متعددة الأبعاد،



نوع البيانات سجل RECORD، التعريف والتصريح عن السجلات، التعامل مع حقول السجل، السجلات المتداخلة، مصفوفة السجلات، التعامل مع مصفوفة السجلات،

المجموعات SETS، إنشاءها وإسناد القيم لها، العمليات على المجموعات (الاتحاد، التقاطع، الفرق)، مقارنة المجموعات، اختبار الانتماء العبارة IN، عمليات القراءة والكتابة.

- **التعامل مع الملفات FILE:**

تعريف الملفات، أنواع الملفات، طرق تعريف الملفات في باسكال، إنشاء الملفات، الملفات الخارجية وذاكرة التخزين المؤقتة للملفات، الكتابة في الملفات، القراءة من الملفات، نسخ ونقل الملفات، تعديل الملفات، التعامل مع الملفات النصية.

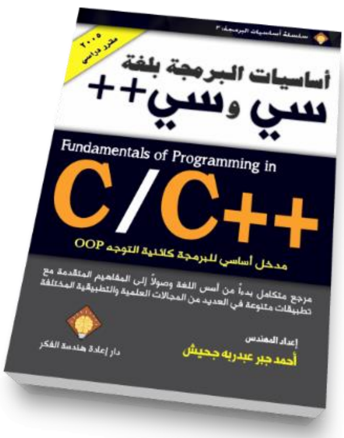
- **المؤشرات Pointers والقوائم المرتبطة Linked Lists:**

التعريف والأهمية، التصريح بالمؤشر، عملية التأشير، فكرة المتغير المرجعي، الإسناد في المؤشرات، إنشاء وحذف المتغير المرجعي،

القوائم المرتبطة التعريف والأهمية، أنواع القوائم المرتبطة، إنشاء قائمة وتدميرها، مثال تطبيقي على القوائم المرتبطة.

أساسيات البرمجة بلغة سي وسي++
:Fundamentals of Programming in C/C + +
مدخل أساسي للبرمجة كائنية التوجه OOP

(٢٠٠٥)



- **الجزء الأول: البرمجة بلغة سي:**
- **أساسيات البرمجة – مقدمة عن لغات البرمجة:**

الحاسب الآلي ومكوناته، لغات البرمجة (لغة الآلة، لغة التجميع، اللغات عالية المستوى)، نبذة تاريخية عن لغة البرمجة سي، البرمجة الهيكلية.

- **مراحل عمل برنامج بلغة سي:**

كتابة البرنامج في المحرر، المعالجة القواعدية – مرحلة ما قبل الترجمة، مرحلة الترجمة، مرحلة الربط، مرحلة التحميل والتنفيذ، مقدمة في البرمجة بلغة سي – البرنامج الأول .. طباعة جملة.

- **المكونات الرئيسية (الهيكل الأساسي) في برامج لغة سي:**

الجزء التعريفي (جزء موجهات المترجم، موجهات الضم (الموجه #include<>‏ والملفات الرأسية)، موجهات التعريف (الموجه #define const1 value)، جزء الإعلان)، الجزء الوظيفي (التوابع (الدوال) والإجراءات، العبارات statements واستخدام الرمز (;))، التمييز بين الأحرف الصغيرة والكبيرة، التعليقات /* ... */ – كيف توضع في البرنامج، تنسيق البرنامج.

- **المتغيرات (المتحولات) Variables:**

شروط التسمية، طرق الإعلان (التعريف) عن متغير، تعريف مجموعة متغيرات معاً، موضع الإعلان عن المتغيرات، أنواع البيانات القياسية (المتغيرات الصحيحة (int, long)، المتغيرات الحقيقية (float, double)، المتغيرات الحرفية (char)، بناء متغيرات سلسلة حرفية (string)، الثوابت (الرقمية – الحرفية)، طرق إسناد (تعيين) القيم للمتغيرات، الإسناد عند الإعلان.

- **العمليات على المتغيرات:**

العمليات الحسابية (+ – * / %) – برنامج .. جمع عددين، أولويات العمليات الحسابية ((),*/%،+،–)، عمليات الإسناد المختصرة (+=،–=،*=،/=،%=)، عمليات الزيادة والإنقاص بواحد (x++, ++x, x--, --x).

- **الشاشة ولوحة المفاتيح:**

أولاً: الكتابة على الشاشة: إرفاق مكتبة الإدخال والإخراج القياسية stdio.h، **كتابة حرف واحد على الشاشة** putchar, putchar، **كتابة مجموعة من الأحرف** puts، **الإجراء** printf **(الأحرف غير المطبوعة** (\b\ t\ n \"' \\)، **طباعة محتويات متغيرات، استخدام أحرف البدل** (%i; %d, %o, %x, %f, %s, %c)، **التحكم في عرض النتائج:** %d٥ – %٥٠5s-% - d **ناحية اليسار).**

ثانياً: القراءة من لوحة المفاتيح: قراءة حرف واحد getchar، **استخدام** getche، getch **من المكتبة** conio.h، **قراءة مجموعة من الأحرف** gets، **استخدام** strlen، **الإجراء** scanf **(نوع البيانات – الأحرف البديلة، مكان تخزين المدخلات، استخدام الرمز** & **والعناوين في الذاكرة، استخدام إجراء** scanf **مع السلاسل والمسافات – أسماء السلاسل عناوين في الذاكرة، إفراغ الذاكرة الوسيطة الخاصة بلوحة المفاتيح** fflush(stdin)، **القراءة المتعددة باستخدام** scanf.

- **بنى التحكم واتخاذ القرارات (الحلقات والشروط):**

الخوارزميات – لغة الخوارزميات، بنى التحكم – التنفيذ أو البنية التسلسلية – نقل التحكم، عمليات اتخاذ القرار: عمليات المساواة والمقارنة (==، !=، >، >=، <، <=)،

البنية الشرطية (if)، **البنية الشرطية** (if else)، **تداخل البنى الشرطية** (if else) – **استخدام البنية** (else if)،

بنية الشرط المتعدد (switch/default) – (switch)،

العامل الشرطي، المعاملات المنطقية (&&، ||، !) – **العمليات المنطقية،**

البنية التكرارية (for)، **البنية التكرارية** (while)، **البنية التكرارية** (do/while)،

التعليمتان break, continue،

أسلوب البرمجة المهيكلة – قواعد تشكيل برامج مهيكلة،

التعليمة المحظورة goto، **نوع البيانات التعدادية** (Enumerated Data Type).

- **التوابع (الدول)** Functions:

التوابع الجاهزة – مكتبة التوابع الرياضية (math.h)، **المكتبات المعيارية والملفات الرأسية، التصريح عن التوابع، تعريف التوابع – دمج التصريح والتعريف، استدعاء التوابع، تمرير المعاملات** (Arguments – **الوسطاء) إلى التوابع (التمرير بواسطة الثوابت، التمرير بواسطة المتغيرات)، الحصول على قيم مرجعية من التوابع – الأمر** (return)، **المعاملات المرجعية** (Reference Arguments)، **المتغيرات و أصناف (صفوف) التخزين – عمر المتغيرات – مجال الرؤية (المتغيرات الخارجية، المتغيرات الآلية، المتغيرات الساكنة).**

- **المصفوفات Arrays:**

المصفوفات – التصريح عن المصفوفات، إدخال / إخراج قيم لعناصر المصفوفة، تهيئة المصفوفة، المصفوفات ثنائية البعد – المصفوفات متعددة الأبعاد، تهيئة مصفوفة ثنائية البعد، تمرير المصفوفات كمعاملات للتوابع.

- **سلاسل الحروف Strings:**

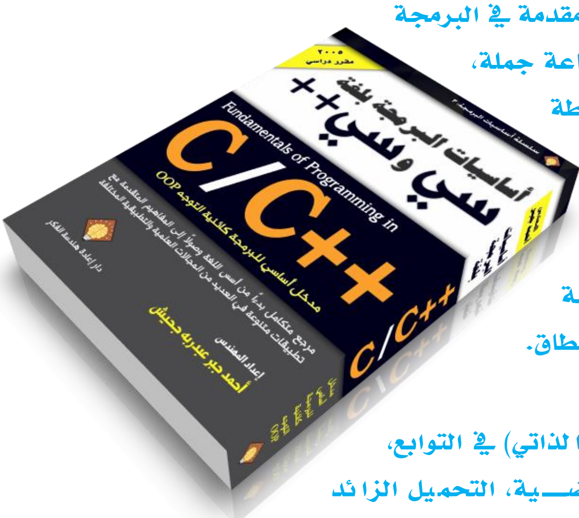
السلاسل الحرفية – متغيرات السلاسل الحرفية، ثوابت السلاسل الحرفية، التعامل مع الفراغات، التعامل مع الأسطر المتعددة، مصفوفة من السلاسل الحرفية، توابع مكتبة التعامل مع سلاسل الحروف ((string.h) توابع نسخ السلاسل (strcpy, strncpy)، توابع إضافة السلاسل (,strcat strncat)، توابع مقارنة السلاسل (strcmp, strncmp)).

- **المؤشرات Pointers:**

العناوين والمؤشرات – العنونة المباشرة وغير المباشرة للمتغيرات، التصريح عن المؤشرات كمتغيرات وإعطاءها قيماً ابتدائية، عمليات التأشير، الوصول إلى المتغير المشار إليه، استدعاء التوابع بالعنوان، تعابير المؤشرات والمؤشر الحسابي، المصفوفات والمؤشرات - العلاقات بين المؤشرات والمصفوفات (تمرير المصفوفات، ترتيب عناصر مصفوفة، المؤشر إلى سلاسل حرفية، السلاسل الحرفية كمعاملات للتوابع).

- **الجزء الثاني: البرمجة بلغة سي++:**
- **مقدمة عن البرمجة بلغة سي++:**

نبذة تاريخية عن لغة البرمجة سي ++، مقدمة عن البرمجة غرضية التوجه OOP، المكونات الأساسية لبيئة عمل سي ++ قياسية، مقدمة في البرمجة بلغة سي ++ – البرنامج الأول .. طباعة جملة، التعليقات باستخدام //، الإخراج بواسطة التابع <<cout، الإدخال بواسطة التابع >>cin، سلاسل الهروب (\n \a \r \t \\ \"’)، المؤثرات (endl, setw) من iomanip.h، العملية الأحادية لتحديد مجال الرؤية (::) النطاق.



- **أفكار متقدمة:**

العودية Recursion (الاستدعاء الذاتي) في التوابع، التوابع السطرية، المعاملات الافتراضية، التحميل الزائد للتوابع، القوالب – قوالب التوابع، استخدام الواصف (const) عند

مع معاملات التوابع، مصـــفوفات الفرز (طريقة الفرز الفقاعي)، مصـــفوفات البحث (البحث الخطي، البحث الثنائي)، استخدام الواصف const مع المؤشرات، المؤشرات على التوابع.

- **أسس ومبادئ البرمجة الكائنية التوجه:**
- **مفاهيم أساسية في البرمجة الكائنية** Object-Oriented Programming-OOP:

التجريد، الصف، الكائن، الخصائص، الوظائف، التغليف، إخفاء البيانات، الوراثة، تعدد الأوجه.

- **الصف** Class:

إنشاء الصفوف، تعريف خصائص الصف ووظائفه، الوصول لمكونات الصفوف، محددات الوصول إلى المكونات (العام public والخاص private)، وظائف التشـــييد constructor، وظائف التدمير destructor، العبارة this، وظائف الإســـناد set والإرجاع get، الوظائف الســـكونية static في الصفوف، الوظائف الصديقة friend.

- **الوراثة** inheritance:

المفهوم والأهمية، توريث الصفوف، محدد الوصول المحمي protected، الوراثة والتشييد والتدمير.

أساسيات نظام التشغيل لينكس Linux

(٢٠٠٥)



- **التعرف على نظام لينكس Linux:**

  ماهيته، نظرة تاريخية، أشـــهر التوزيعات Distribution، طرق الاختيار بين التوزيعات.

- **تجهيز النظام للعمل على جهازك:**

  كيفية الحصــول على التوزيعات، التطبيق على نظام ردهات لينكس RedHat Linux الإصـــدار الأحدث 9، و نظام ديبيـان Debian الإصـــدار 3.0R3، وبعض التوزيعـات الشهيرة.

- **التعامل مع سطر الأوامر:**

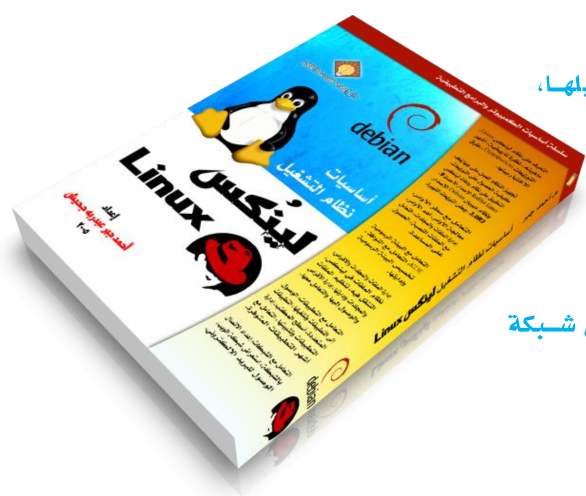
  معـالجـة الأوامر، أهم الأوامر، إدارة الملفـات والمجلـدات، التعامل مع الملفات النصية، الحصول على المساعدة.

- **التعامل مع البيئة الرسومية KDE:**

  التعامل مع النوافذ، تخصيص البيئة الرسومية وإدارتها.

- **إدارة الملفات والمجلدات والأقراص:**

  نظام الملفات في لينكس، التنقل فيه، تنظيم الملفات والمجلدات وإدارتها، إدارة الأقراص والوصــول إليها والتعامل معها.



- **التعامل مع التطبيقات:**

  الوصـــول إلى التطبيقـات وتشــغيلها، التطبيقات المتعددة، أســطح المكتب، إدارة التطبيقات وتثبيتها، التعا مل مع أشهر التطبيقات المتوفرة.

- **التعامل مع الشبكات:**

  إعداد الاتصــال بالشــبكة، اســتعراض شــبكة الويب، الوصول للبريد الإلكتروني.





م. أحمد جبر جحيش

ترقبوا قريباً كتاب
أساسيات
البرمجة كائنية التوجه
باستخدام لغة جافا



هذا الكتاب

يهدف هذا الكتاب إلى توفير مرجع تعليمي منهجي عربي للمبرمجين وخاصة دارسي البرمجة الكائنية المنحى، ويسعى إلى تغطية مجموعة من المواضيع الأساسية؛ والتي يمكن إيرادها في الآتي:

- المفاهيم الأساسية في لغة جافا، مع مدخل مفاهيمي للبرمجة كائنية المنحى
- بناء وتطوير تطبيقات جافا ومقدمة إلى بريمجيات جافا
- نبذة مختصرة عن استخدام المحرر Kawa
- التعامل مع خرج البرامج، وحجز المواقع في الذاكرة والمتغيرات
- طرق الإسناد وإجراء العمليات الحسابية وعمليات المقارنة وعمليات الزيادة والإنقاص، والعمليات المنطقية
- أساسيات الرسم في الجافا
- التعرف على الخوارزميات ولغتها (الشفرة الزائفة) Pseudocode
- نظرة شاملة حول بُنى التحكم الأساسية (بُنى الاختيار والتكرار)، والتعليمتان break وcontinue وإمكانياتهما
- أسلوب البرمجة المهيكلة Structured Programming
- نبذة مختصرة عن لغة النمذجة الموحدة UML
- استخدام الطرق والتوابع الجاهزة والطرق الرياضية المضمنة
- بناء وتطوير الطرق والتوابع الخاصة
- قواعد مجال الرؤية Scope Rules، والتحميل الزائد للطرق Method Overloading، والاستدعاء الذاتي (العودية) Recursion

أساسيات البرمجة بلغة
جافا
Fundamentals of Programming in
Java

دار إعادة هندسة الفكر